Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D7.1

Initial report on benchmarking of management protocols

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:
Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France
Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Initial report on benchmarking of management protocols

Type: Public

Editor(s): Radu State

E-mail: state@loria.fr

Author(s): Laurent Andrey, Juergen Schoenwaelder, Aiko Pras, George Pavlou, Krzysztof Nowak

Doc ID: D7.1-v2.4.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V1.1	October 1, 2006	Radu State	First version, providing the ToC
V2.1	October 7 2006	Radu State	First complete draft version
V2.2	November, 4 2006	Radu State	Added section 2
V2.3	December 12, 2006	Radu State	Final editing
V2.4	April 10, 2007	Olivier Festor	Editor changes

Legal Notices

Table of Contents

(This page is left blank intentionally.)

1 Executive summary

Management performance evaluation means assessment of scalability, complexity, accuracy, throughput, delays and resource consumptions. In this deliverable, we focus on the evaluation of management frameworks. We describe generic requirements for scalability in section 2. The next sections address specific management frameworks like NetConf, SNMP, Web services and JMX. For each management framework, we define specific benchmark approaches and analyze the results obtained using this benchmarking approach. This deliverable is based on the three milestone publications [26, 27, 28] made by the members of this workpackage.

2 Introduction

2.1 A pragmatic view on management scalability

As networks grow in time with respect to number of nodes, available throughput and complexity of nodes themselves, the need for flexible, efficient and scalable network management systems becomes more and more distinct. Within each managed system the amount of data required to keep the network running is proportional to the complexity of the system itself. The more data is required, the more data needs to be sent through the network to Network a Operations Center (NOC), consuming more bandwidth and processing power in NOC. This simple statement outlines basic scalability issues encountered by network operators. Scalability requirements are directly related to management models in use and vice-versa, a management model is selected with respect to scalability requirements.

The most deployed model is a centralized approach to network management, where the entire network is monitored by a single NOC. There operators can easily see the current snapshot of the entire network state. In this model, the entire data processing is being performed within the NOC while managed nodes merely collect statistics and report occurrences of specified events to the NOC acting as the manager. Such a management approach is the most simple to implement, as it is based on a simple manager-agent architecture. The model works well for relatively small network environments, where the number of supervised devices is small, just as is the amount of information transferred between agents and the NOC. This kind of system is the optimal choice when most management activities require full knowledge of the network state and where managed devices have limited computational power (or other resources) and cannot process management data themselves. Starting with this basic model, we can derive general management scalability requirements by describing the most important problems one can encounter in its deployment in large networks.

As the bandwidth requirements of applications grow (especially because of increasing popularity and use of multimedia content, such as streaming audio / video), network operators try to satisfy customers by expanding their network backbones. Existing links need to be updated at higher rates to make it possible to achieve greater transmission rates. Some new links are introduced for the sake of load balancing. This involves deployment of additional hardware and the extending of the management infrastructure. In order to analyze the data according to centralized management paradigm, information must be periodically requested by NOC from each device to keep the network picture up-to-date; the devices need to report any abnormal occurrences as well. This creates another network load increase. Such scenario clearly shows how ineffective bandwidth usage gets when the network grows, especially when core network is already heavily overloaded. In order to reduce amount of information that needs to be sent over the network purely for management purposes one needs to switch from centralized approach to something different. The solution might be distributed management. This way, some tasks that do need access to description of current state of entire network may be transferred from NOC to managed device. This, however, raises other problems, related to resources availability.

Dedicated management stations are generally equipped with faster processing units, more memory and more storage space than agents in devices scattered throughout the network. Yet modern networking hardware is usually capable of handling extra tasks that might be related to network management, effectively limiting bandwidth usage. This must be taken

into consideration when planning network upgrades within existing systems and when building new ones. Managed nodes must be operated by efficient, high-performance, flexible devices that can take part in network management themselves, as stated in decentralized Management by Delegation paradigm [29]. This method can be used in appliances where computations can benefit from spatial distribution. Of course this is not always possible, therefore in real life most probably both centralized and distributed management will coexist throughout the managed network [30]. Network management frameworks should be aware of all the issues related to that and should make such coexistence possible and as effective and easy to use as possible.

NOCs should use modern, flexible frameworks capable of running at several detail levels. As the network grows, eventually amount of managed devices gets too large to effectively operate all of them as planned, since single operator can no longer grasp the complete network state. Therefore there is need for a multilevel scheme in which devices and tasks can be put together in groups of similar importance and relevance for the purpose of increasing overall comprehensibility of the system. This also addresses the limited computational power issue – operations on aggregated data should be faster than complex processing of all data / commands related to each and every device deployed in the network.

Another issue – at least as important as the one described in previous paragraph – is the choice of communication protocols used between devices and the NOC – and between devices themselves in case of distributed management. From the point of view of network operator it is essential to use standardized, “open” protocols for data transfer, as it prevents vendor dependency which could lead to financially disadvantageous situation or even inability to expand network (in case vendor encounters any problems related to delivering appropriate hardware). Keeping network free of proprietary protocols makes it also easier to upgrade in case current protocols prove not to be scalable or otherwise inefficient or unsuitable for future purposes. It also makes “in-house” development of management solutions possible if no suitable software or hardware already exists on the market.

Communication protocols selected need to be capable of transmitting data not just between the NOC and managed devices, but also between managed devices themselves, thus enabling the whole management system to work according to distributed management scheme. Such protocol should introduce as little overhead as possible, as this is essential in large scale commercial applications, where each and every bit unrelated to non-management traffic might be considered a financial loss, like in case of e.g. high-load links on the core network of large (national / international) operator. Protocols used also need to be general enough, so that users can transmit virtually any kind of data with it. This might become useful when implementing distributed management, which requires communication between managed devices. Such hardware may require completely different data types exchange in the future when new solutions are deployed throughout the network.

Just as protocols need to be standardized to avoid being dependent on single management solutions vendor, the network management software itself should be designed to run on multiple platforms if possible, as this further expands upgrade possibilities and prevents the dangerous effect of vendor lock-in. This requirement might be considered optional in case the software communicates with devices using standardized protocol, since inefficient software may then be replaced together with the underlying system, however this should be avoided as it might greatly inflate upgrade costs. In any case, when choosing network management software one should pay attention to platform dependency of considered software.

Scalability matters especially in case of large network operators. Let's consider as an example Poznan Supercomputing and Networking Center, which is responsible for PIONIER NREN operation and management, as well as POZMAN metropolitan network and PSNC LAN. This results in a relatively big number of hardware being managed by PSNC. PIONIER itself consists of 40 managed nodes. Metropolitan area network consists of another 68 nodes. Together with PSNC LAN (27 devices) we achieve a total number of 135 nodes. Within the management process, there are 3710 statistics collected at the NOC (1138, 1642 and 930 related to PIONIER, POZMAN and PSNC LAN respectively). Those include number of incoming / outgoing octets, sometimes number of packets and sometimes temperature monitoring. The whole system consists of 3548 logical network interfaces. Although roughly estimated mean traffic to / from PIONIER management station is currently about 10 kbit/s (not including statistics gathering, which consumes another dozen of kbit/s), it is clear that this structure requires efficient management and thorough planning of future expansion. It is essential to take scalability into consideration in this process.

2.2 Outline

In this deliverable, we focus on the evaluation of management frameworks. We describe generic requirements for scalability in section 2. The next sections address specific management frameworks like NetConf (section 3), SNMP (section 4), Web services (section 5) and JMX (section 6). A summary of the presented work is given in section 7.

3 NetConf benchmarking

3.1 Netconf architecture

NetConf is a XML-based network configuration protocol, proposed by the IETF in order to achieve simplicity and management data hierarchy. Netconf is stream-oriented and relies on a Remote Procedure Call (RPC) mechanism. The client (a Netconf manager) typically issues a request to a server (a Netconf agent) which returns a reply rpc-reply. An rpc tag always embeds a Network Management operation which can be one of get, get-config, edit-config, copyconfig, lock, unlock, close-session or kill-session.

An important point is that a manager sees the configuration data as a XML document, thus giving him a very high, uniform, easy and clear abstraction level. The operations are conceptually performed on that virtual or real document. No other communication channel is even needed since all services can potentially be configured through Netconf.

Before sending rpc operations, a manager must first establish a Netconf session on the agent. A session is carried over a TCP connection and remains open until a close-session or kill-session is issued to the agent. In order to initialize that session, each party sends a hello message containing its capabilities and a session-id is issued by the agent. This message is exchanged asynchronously over the TCP connection. Capabilities are a feature that allows each party to advertise its supported features (data model, vendor-specific operations, etc...). Each capability is identified using a unique URN identifier. Netconf defines a set of capabilities like, for instance, #xpath, #startup, #candidate that can be extended with new ones.

Get and get-config operations allow to retrieve data from the Netconf agents and come with two possible filtering methods. While the first one, subtree filtering, is specific to Netconf, the second one, XPath, is an independent technology.

Each method has its interest. XPath allows to build fine-grained request like /netconf/system/rpm-list/rpm[contains(text(),'java')] which selects all rpm elements that contain java into their text value. It is also possible to select the number of output nodes and the text value at different XML level using multiple criteria. Subtree filtering allows to build a kind of template document that is filled by the agent. It is legitimate to wonder if one of the methods is performing better than the other, and this question has been actively debated in the IETF working group (see https://psg.com/lists/netconf/). We have performed a series of experiments on this issue and the results are discussed in this paper.
The current IETF specifications represent a pragmatic solution for XML based network management, making for commonly agreed on standard. We extended the IETF proposed specifications with more advanced features related to security, efficient data transmission and agent site filtering and scoping functionalities.

Extending the Netconf protocol with advanced security features is of crucial importance in a multi-party configuration environment, where large-scale infrastructures are managed by multiple administrators having different competencies and security clearance levels. The Integrated Security Framework (see Figure 3.1) shows our integrated security components. The core is split into four main security areas: authentication, integrity, confiden-

tiality and access control.

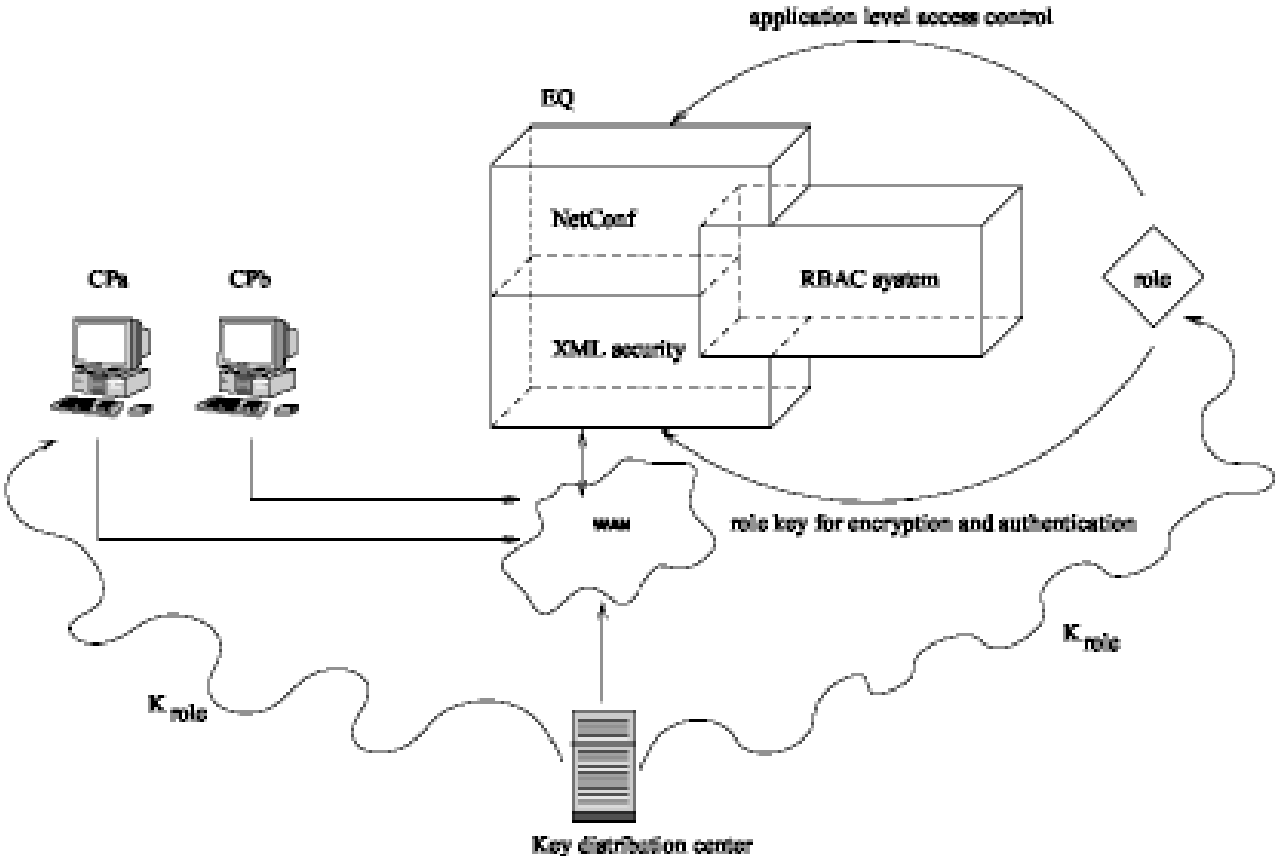

Figure 3.1: Security architecture

Each area is related to one or more adjacent techniques or concepts. For instance, XML-Encryption is dedicated to confidentiality. Access control means a flexible mechanism to manage access to resources. The flexibility aspect is evaluated regarding the platform dynamicity and size. While dynamicity implies the capability to frequently modify the access rights, the size is dependant on the number of devices and managers. Authentication is the mechanism that guarantees that an entity is the one it claims to be. Integrity is the mechanism that ensures that the management data in transit has not been modified. Already defined XML security paradigms (XML-Signature [1], XMLEncryption [2]) are used to perform data access control. The security model for Network configuration is built around several main entities depicted on Figure 5. While configuration providers (CP) act like a data configuration source, the managed devices (EQ) run a Netconf agent.

In order to address an efficient security plane, we propose a distributed Role Based Access Control type mechanism, in which roles are distributed on the fly. However, access is performed in a non-intermediated way, meaning that the architecture delivers tokens (confidentiality and authentication keys) to principals. To prove the ownership of tokens, an entity signs and cyphers the messages with the keys bound to the roles. In order to manage this dynamicity, we need a fast mechanism for the key exchange system. This is particularly obvious if role revocations are considered. For instance, CPa and CPb can load different partial configurations onto an agent configuration making for a multi-source configuration. These configurations are loaded with respect to different access rights; an agent

must be able to decrypt incoming configurations and to bind them to some access rights. We will describe in this section the mechanisms needed to meet these requirements.

An RBAC manager is responsible for security mediation among configuration providers and managed devices. This entity provides the different security services: authentication, data integrity, confidentiality and access control. The RBAC manager hosts RBAC policies which are dynamically deployed on our network entities.

RBAC allows high level and scalable access control process and configuration. The RBAC model consists in a set of users, roles, permissions (operations on resources) and sessions. The originality of the RBAC model is that permissions are not granted to users but to roles, thus allowing an easy reconfiguration when a user changes his activity. A role describes a job function or a position within an organization. The RBAC model allows the description of complex access control policies while reducing errors and administration costs. Introduction of administrative roles and role hierarchies makes it possible to considerably reduce the amount of associations representing permissions to users allocation.

All agents store an RBAC policy describing the permissions that a role is allowed to perform. In this model, the managers send their requests on behalf of a role. Therefore an agent is able to authenticate and decrypt the requests because it knows the role in use. For each role, the architecture provides one key for authentication and one key for encryption.

After the role authentication and decryption, an agent can start the access control step. According to its local policy and to the role used, it can decide if the request is allowed or not.

As XML language introduces a potentially huge verbosity (see Figure 4), use of compression can dramatically save network bandwidth consumption. Moreover, it gives a simple way to avoid clear text transmissions.

Using compression is of interest, in particular in combination with security features. Since processing time for encryption increases fast with the message size, compression before encryption can reduce encryption processing cost. These issues are illustrated and discussed on the basis of the performance tests presented in the next section.

3.2 Experimental performance evaluation

To validate and evaluate the proposed security architecture and the compression module, a prototype has been implemented. The benchmarking tests are done locally on a Pentium 4 3.2GHz with 2 Go RAM (see Table 3.1).

In order to give an overview of the global performances, the agent can manage up to 17 get-config requests per second when no logging mechanism and no security is deployed. When security (encryption, RBAC) and compression are enabled, the performance decreases to 8 requests per second. For the benchmarking, we used only get-config operations. These represent the worst case for security performance because they require a low CPU consumption. If the tests are done on complex requests, the CPU consumption dedicated for security will be negligible compared with the time to perform the Netconf request. With fast requests, it is possible to highlight the influence of security mechanisms on the global processing time. However, we compare compression and encryption performances for both large as well as small sized XML documents.

Description	Characteristics
RAM memory	2 Gigabytes
CPU	Pentium 4, 3.2 GHz
OS	Linux FC 4, kernel 2.6.12
Python	2.4
security libraries	PyXMLSec, paramiko
crypto. keys length	128 bits

Table 3.1: Test environment characteristics

Figure 3.2 illustrates different use cases:

1. encryption + compression
2. encryption,
3. compression,
4. default.

The measurements are done within the agent. For this test, 7 different getconfig requests, applying either subtree filtering or XPath, are performed. Each of them is evaluated 64 times and an average of the processing time is computed. This average is represented on the figure. Each column relates to the total processing time of a get-request.

The results show that compression and decompression time is insignificant compared with the global request processing time. Also RBAC processing takes roughly 6% of the global time. Encryption represents 9.5% and decryption 3.6%. Using cryptography, compression and access control consumes about 20% of the global time.

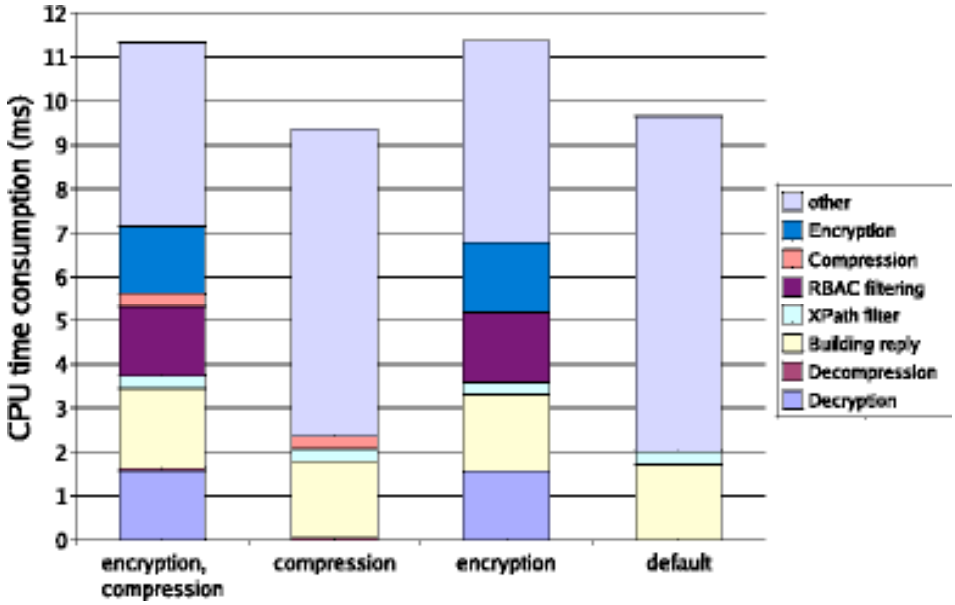

Figure 3.2: Global request processing time

Note that the total processing time increases by a factor 60 if a XML Schema validator systematically checks the received messages with the Netconf XML schema. The total average processing time is changing from 11 ms to 650 ms. In such a case, the proportion of CPU time dedicated to compression and security decreases. CPU or memory consump-

tion due to XPath technology is of minor importance compared with the use of a XML schema validator. Filtering with XPath takes less then 1 ms. Checking the validity of a Netconf message with XML Schema takes more than 600 ms. These results give an overview of the global performances of this implementation as well as the performance of each functionality relatively to the others.

Table 2 displays the result response time for different typical data retrieval tests. The agent and the manager are located on two devices connected to a hub and communicate over SSH. For this test only, the agent is located on a centrino laptop 1.6GHz with 1 Gigabytes of memory and running Fedora Core 4. The global request processing time per agent remains the same as the previous results, but on the manager side, the results are as follows. The manager is sending 1000 get-config requests using XPath capability and then, the average response time is computed for each. The tested BGP configuration is realistic enough to be used in a real network. While a test on a sample BGP configuration gives a response time of 58.40 ms, a realistic BGP configuration is retrieved in around 200 ms. These results assess the suitability of Netconf for such tasks.

Management data	Response time
network interfaces	38.73
network routes	49.71
BGP configuration	200.08
RBAC policy	43.05

Table 3.2: Average response time in ms

To conclude on the use of compression, we observed that bandwidth is used efficiently and CPU time consumption is not significant when compared to the global processing time.

3.3 Using both compression and encryption

An experiment comparing the processing time of compression, encryption and compression+encryption on XML documents having different sizes led us to the following conclusion: when the size of the document reaches a threshold, it is faster to perform first compression and then encryption than to only encrypt the document. It means that beyond this threshold, using both compression and encryption allows not only to save network bandwidth but also to decrease the agent CPU consumption at the same time.

3.4 Performance of XPath versus subtree filtering

In order to compare the performances of both node selection methods, XPath and subtree filtering, we prepared 8 get-config requests with subtree filtering and their equivalent XPath based requests. The requests are available in http://www.loria.fr/vcridligv/-xpathSubtree.html. We used as many different options as possible in subtree filtering requests: selection nodes, content match nodes.

Each request is sent 10 times and then an average time is computed. This experiments showed how subtree filtering is adapted to build a customized document and how XPath is efficient to select nodes on more complex criteria (for instance, conditions on text node value or node number).

Subtree filtering is somewhere between XPath and XSLT: it allows to select nodes but also to apply a mask to the configuration. It is always possible to translate a subtree filtering request to XPath but the philosophy is different and both methods are useful and complementary.

The results displayed on Table 3.3 show that subtree filtering and XPath have similar average processing time. Each request is made 1000 times and an average is computed afterwards. The two filtering approaches are slightly different. When XPath is adapted to select isolated data items, subtree filtering is more suitable to build complex views made of different parts of the config and to visualize them previously in a template-like document. In order to build a document made of completely different parts with Xpath, a manager has to use a | symbol to join the various parts of the selected nodes. However, the subtree filtering can only use absolute addressing while XPath allows both relative and absolute expressions like, for instance, //iface[name='eth0'] or /netconf/network/interfaces/iface[name='eth0']. The advantages of XPath are 1) it consumes less bandwidth since the requests are much more compact than subtree filtering and, 2) it does not require an important effort from the managers.

To weight these results, it is important to note that the total request processing time is much higher than the filtering processing time, what ever be the selection method. To conclude on filtering, both approaches have their advantages and are complementary.

Filtering	1	2	3	4	5	6	7	8	9	10	11	12
Subtree	1171	1482	694	463	477	1503	2114	1371	919	775	1733	863
Xpath	1173	1039	1219	194	687	1286	2828	1973	852	916	1332	2368

Table 3.3: Performances of node selection methods in µs

3.5 RBAC model benchmarking

We have also assessed the impact of access control on CPU time consumption. We implemented a RBAC policy with XPath as a resource addressing scheme. In our model, a Netconf request is sent on behalf of a role. The agent is able to build the list of permissions for that role and all its junior roles. The model implements a role hierarchy: a role inherits the permissions of all its junior roles. The results, illustrated on Figure 3.3, show the additional processing time needed for the access control mechanism. These values also outline the effect of access control mechanism and the number.

When running over SSH, the total time between the emission of the first request and the reception of the last response is 16.77 seconds. When running over XML-Encryption, it takes 15.62 seconds. Therefore, processing times are quite similar

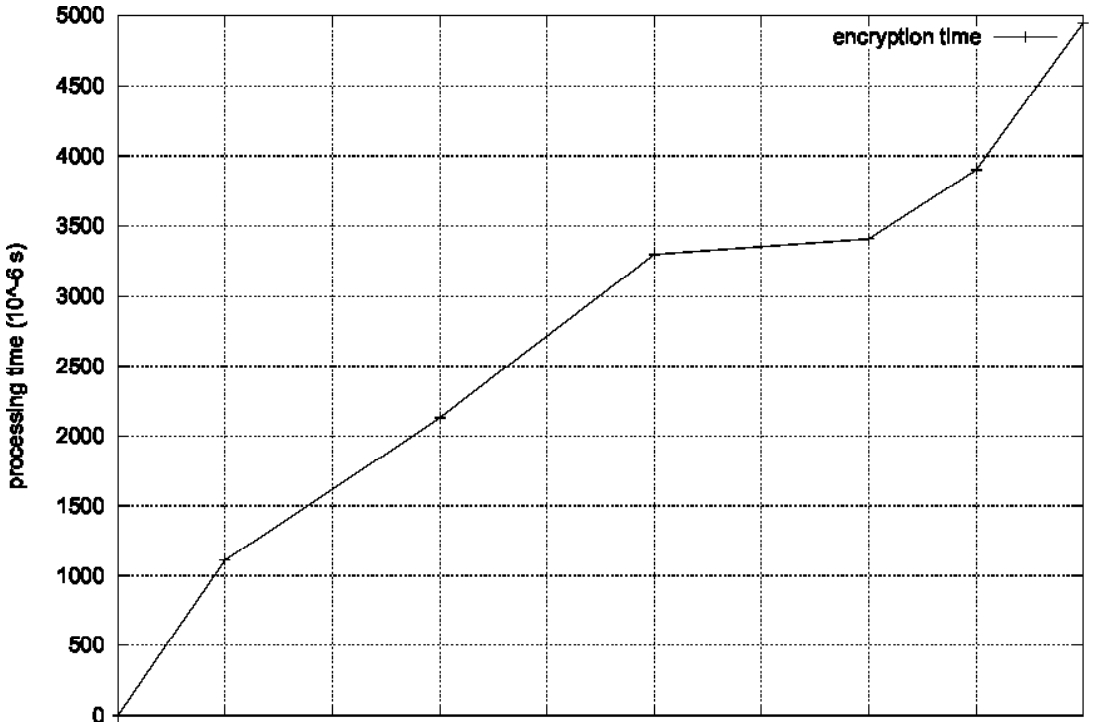

Figure 3.3 : RBAC performance evaluation

4 SNMP performance evaluation

Network devices maintain large amounts of management data. This data can help to provide insight as to how the network is performing or which abnormal events have been observed. Moreover, management data can be used to understand how a device is configured and to change the configuration of that device. The Simple Network Management Protocol (SNMP) [3] allows both for management data to be collected remotely from devices and for devices to be configured remotely. It was first published in August 1988 and since then it has been widely used in network management.

There was no security implemented in SNMP version one (SNMPv1) and the attempts to add security to SNMP version two (SNMPv2) lead to failure as well. Version 3 of the Simple Network Management Protocol (SNMPv3) added security to the previous versions of the protocol by introducing a User-based Security Model (USM) [4]. The USM was designed to be independent of other existing security infrastructures, to ensure it could function when third party authentication services were not available, such as in a broken network. As a result, USM utilizes a separate user and key management infrastructure.

Network operators have reported that deploying another user and key management infrastructure introduces significant costs and hence the USM design is actually a reason for not deploying SNMPv3. To address this issues, a new security model is currently being defined by the Integrated Security Model for SNMP (ISMS) working group of the Internet Engineering Task Force (IETF) which leverages the Secure Shell (SSH) [5] protocol.

It is designed to meet the security and operational needs of network administrators, maximize usability in operational environments to achieve high deployment success and at the same time minimize implementation and deployment costs to minimize the time until deployment is possible.

The SNMP architecture [6] was designed to be modular in order to support future protocol extensions such as additional security models. The architecture defines several subsystems and interfaces between subsystems that should remain unchanged when subsystems are extended. The goal was to reduce side effects that can occur without such an architectural framework when the protocol is extended.

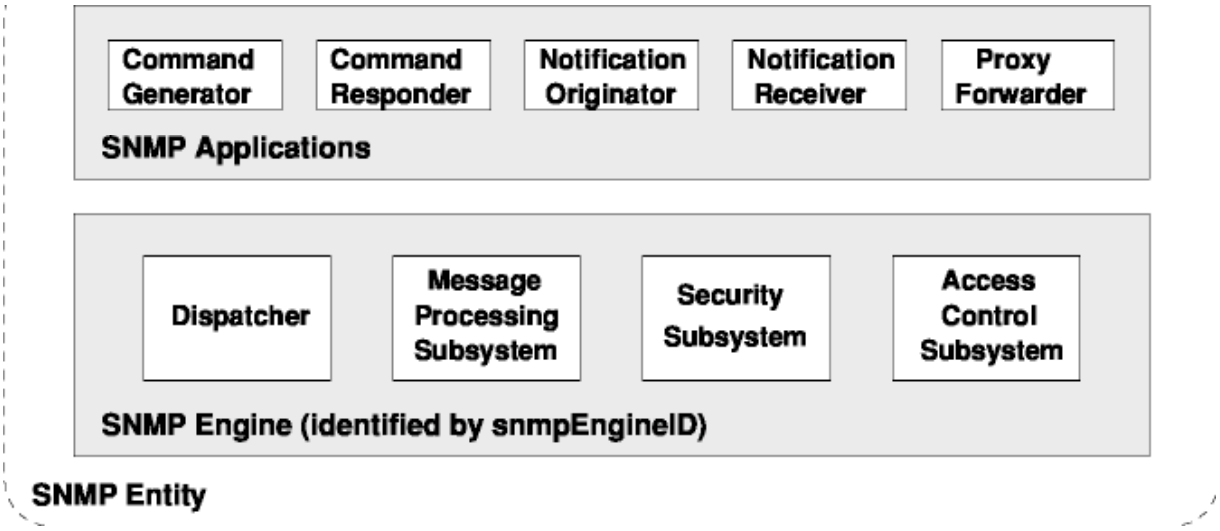

Figure. 4.1. Structure of an SNMP entity according to the SNMPv3 architecture. Every subsystem of an SNMP engine may

contain multiple models while there is only a single dispatcher. A set of SNMP applications together with the SNMP engine form an SNMP entity.

According to the SNMP architecture, an SNMP engine consists of a message processing subsystems, a security subsystem, an access control subsystem, and a single dispatcher (Fig. 4.1). Each subsystem can contain multiple concrete models that implement the services provided by that subsystem. The interfaces between subsystems are defined as Abstract Service Interfaces (ASIs). The dispatcher is a special component which controls the data flow from the underlying transports through the SNMP engine and up to the SNMP applications1.

As of today, most SNMPv3 implementations support three message processing models for SNMPv1, SNMPv2c, and SNMPv3 and two security models, namely the User-based Security Model (USM) (used by the SNMPv3 message processing model) and the Community-based Security Model (CSM) (used by the SNMPv1 and SNMPv2c message processing models). There is only a single View-based Access Control Model (VACM) so far.

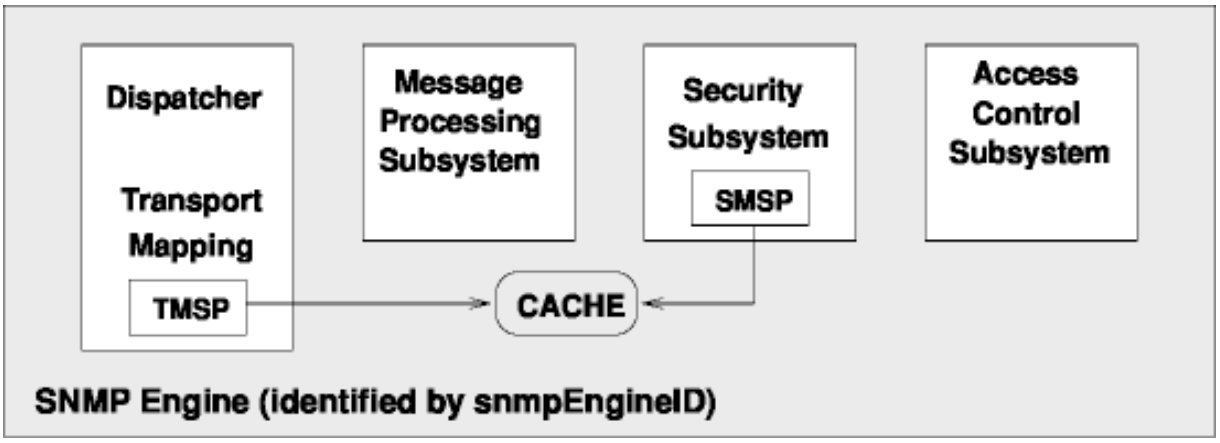

Figure. 4. 2. Structure of an SNMP engine that supports transport mapping security models. The Transport Mapping Security Processor (TMSP) and the Security Model Security Processor (SMSP) communicate via a shared cache.

The design of the SNMP architecture assumes that security services (authentication, data integrity checking, encryption) are provided as part of the SNMP processing. If, however, the security services are provided by the transport over which SNMP messages are exchanged, the architecture does need some extensions. The approach followed by the ISMS working group of the IETF [5] is to split a transport mapping security model (TMSM) into two parts (Fig. 4.2):

1. The Transport Mapping Security Processor (TMSP) [7] is the portion that is part of the message transport and that performs the actual security processing.
2. The Security Model Security Processor (SMSP) [8] is the portion that realizes the appropriate security model required by the SNMPv3 architecture.

Transport security protocols are typically session-based. They usually have a session establishment phase where a session key and some shared state is established followed by the secured data exchange. This is very different from the message-based approach used by USM where all security information is carried in every single mes-

sage exchanged between two SNMP engines and there is no notion of a session or a session key.

4.1 SSH Security Model for SNMP

The Secure Shell (SSH) protocol [5] is a protocol for secure remote login and other secure network services over an insecure network. It consists of three major components:

The Transport Layer Protocol provides server authentication, confidentiality, and integrity. It may optionally also provide compression. The transport layer protocol typically runs over a TCP/IP connection, but might also be used on top of any other reliable data stream. It uses public-key cryptography to authenticate the server to the client and to establish a secure connection which then uses a session key and a symmetric encryption algorithm to protect the connection.

The User Authentication Protocol authenticates the client-side user to the server. It runs over the transport layer protocol. SSH supports several different user authentication mechanisms such as password authentication, publickey authentication, and keyboard-interactive authentication (which supports challenge-response authentication mechanisms).

The Connection Protocol multiplexes the encrypted connection into several logical channels. It runs over the transport layer protocol after successful completion of the user authentication protocol. Every channel has its own credit-based flow control state in order to deal with situations where channels are connected to applications with different speeds.

Note that SSH authentication is usually asymmetric: An SSH server authenticates against an SSH client using host credentials (host keys) while the user authenticates against the SSH server using user credentials (user keys or passwords).

The SSH Security Model (SSHSM) for SNMP [8] is an instantiation of a TMSM. The specification details the elements of procedure for the TMSP and the SMSP portions of the SSHSM. It also deals with details such as engineID discovery and the handling of notifications. Notification delivery is not straight forward due to the asymmetric authentication provided by SSH and the requirement to exercise access control in a consistent way for read, write, and notify access. The details are still being discussed in the ISMS working group of the IETF at the time of this writing.

With the SSHSM, no security parameters are conveyed in SNMPv3 messages. Accordingly, the msgSecurityParameters field of SNMPv3/SSH messages carries a zero length octet string and the implementation of the security model portion of the SSHSM simply retrieves the necessary information by accessing a cache which is shared between the transport mapping porting and the security model portion of the SSHSM.

4.2 Implementation

The prototype implementation of SNMP over SSH developed at IUB is an extension of the widely used open source Net-SNMP SNMP implementation. For the SSH protocol, the libssh library was used, an open source C implementation of SSH. The libssh library contains all functions required for manipulating a client-side SSH connection and an experimental set of functions for manipulating a server-side SSH connection.

The implementation, at the time of this writing, does not implement the SSHSM as it is currently discussed in the IETF. The prototype basically only supports the TMSP part of

SSHSM plus a slight modification of the Community based Security Model (CSM) which passes the authenticated SSH user identity as the security name to the access control subsystem. This basically gives us SNMPv1/SSH and SNMPv2c/SSH while the ISMS working group defines SNMPv3/SSH.

The implementation itself consists of a new transport module which is in the order of 1200 lines of C code. The Net-SNMP internal API for adding transports worked well and did not require any changes. The fact that Net-SNMP already supports stream transports was convenient. For password authentication, the prototype calls the Linux Pluggable Authentication Modules (PAM) [9] library to make it runtime configurable how passwords are checked.

Most of the development time was spend on optimizing the performance of the implementation since the overall latency initially was surprisingly high. In order to optimize the performance of the SSH transport domain, we investigated the influence of TCP's Nagle algorithm as well as the windowing mechanism of the SSH protocol.

During our initial measurements, we observed that the execution of a snmpget operation over the SSH transport domain required approximately 800ms. This delay was mostly introduced by the Nagle algorithm for TCP — the transport layer was waiting for a certain amount of data to be received from the snmpget application before transmitting it to the agent. We therefore, disabled the Nagle algorithm by enabling the TCP NODELAY flag on the agent and on the manager side of the connection. This lead to a significant improvement in the performance of the SSH transport domain as the time required for the execution of a snmpget operation went down to 56.5ms. We further modified the libssh library and disabled the Nagle algorithm immediately after establishing the TCP connection between the agent and the manager. This further decreased the time required for a snmpget operation to 16.17ms on our fast machines.

The SSH windowing mechanism is used to specify how much data the remote party can send before it must wait for the window to be adjusted. In the OpenSSH implementation such window adjustment messages are only exchanged periodically. During our initial observations we noticed that each message exchanged between the agent and the manager was followed by a window adjustment message. These additional messages introduced significant bandwidth overhead as well as latency overhead for long sessions. As a result the SSH transport domain performed worse than the USM transport domain with respect to latency and bandwidth. In order to optimize the performance, we modified the libssh library to send window adjustment messages only when necessary. This improvement lead to better bandwidth and latency performance of the SSH transport domain when compared to the USM transport domain as explained below.

The performance of our SNMP over SSH prototype has been evaluated by comparing it against SNMPv3/USM with authentication and privacy enabled, running over both TCP and UDP. In addition, to establish a baseline, the performance of plain SNMPv2c over both TCP and UDP was measured. The following sections first describe the experimental setup and then discuss the session establishment overhead and the performance for walks of different sizes without and with packet loss. Finally, the bandwidth used by the different SNMP transports is compared and the memory requirements for keeping open SSH sessions on a command responder is discussed.

4.3 Experimental Setup

The experiments were performed on three Debian GNU/Linux machines (see Table 4.1). The machines were connected via a switched Gigabit Ethernet with sufficient capacity.

Name	CPUs	RAM	Ethernet	Kernel
meat	2 Xeon 3 GHz	2 GB	1 Gbps	2.6.16.14
veggie	2 Xeon 3 GHz	1 GB	1 Gbps	2.6.12.6
	1 Ultra Sparc IIi	128 MB	100 Mbps	2.6.16.14

Table 4.1. Machines used during the measurements

SNMP/USM measurements, we used the authentication plus privacy security level with HMAC-MD5 as the authentication algorithm and AES-128 as the encryption algorithm. The libssh library was configured to also use the HMACMD5 authentication algorithm and the AES-128 algorithm for encryption and null compression.

We instrumented the snmpget, snmpwalk and snmpbulkwalk programs (part of the Net-SNMP package) to measure the latency by calling gettimeofday() before opening a session (but after parsing of MIB files) and after closing the session and computing the time difference. The output was directed to /dev/null and each experiment was repeated 100 times. The tcpdump tool was used to calculate the number of bytes exchanged and the pmap tool was used to measure the memory sizes of the processes. Packet loss was simulated by using the netem network emulation queuing discipline of the Linux kernel on meat and turtle.

The object identifier (OID) used for snmpwalk and snmpbulkwalk measurements was the interface table ifTable [10]. The object identifier used for snmpget measurements was the scalar sysDescr [11]. The number of rows in the ifTable was manipulated by creating virtual LAN (VLAN) interfaces.

Table 4.2 shows the result of performing snmpget requests on the scalar sysDescr using different transports. There is a significant cost associated with the establishment of SSH sessions. This is, however, not surprising since the SSH protocol establishes a session key using a Diffie-Hellman key exchange (using public-key cryptography) before the user authentication protocol is executed and the SSH channel is established. Since the cryptographic operations are CPU bound, the session establishment times increases significantly on our slow test machine.

Protocol	Time (meat)	Time (turtle)	Data	Packets
SNMPv2c/UDP	1.03 ms	0.70 ms	232 byte	2
SNMPv2c/TCP	1.13 ms	1.00 ms	824 byte	10
SNMPv3/USM/UD	1.97 ms	2.28 ms	668 byte	4
SN	2.03 ms	3.03 ms	1312	12
SNMPv2c/SSH	16.17 ms	91.62 ms	4388	32

Table 4.2. Performance of snmpget requests (sysDescr.0)

Table 4.2 also shows that there is an clear difference in the amount of data (total size of the Ethernet frames) and the number of IP packets exchanged between UDP and TCP transports. While this overhead is usually not a big issue on a well functioning local area network, it might be an issue in networks with large delays or high packet loss rates.

Figure 4.3 shows the latency for the retrieval of the ifTable with different sizes using snmpwalk. The difference between the transports UDP and TCP seems marginal compared to the difference caused by enabling authentication and privacy. The performance of the SSH transport is interesting. Initially, the costs of establishing an SSH

channel with a new session key cause the SSH transport to perform worse than SNMPv3/USM. However, there is a breakeven-point after ~ 500 SNMP interactions where the SSH transport becomes more efficient than SNMPv3/USM. This observation seems to be independent of the speed of the machine hosting the command responder.

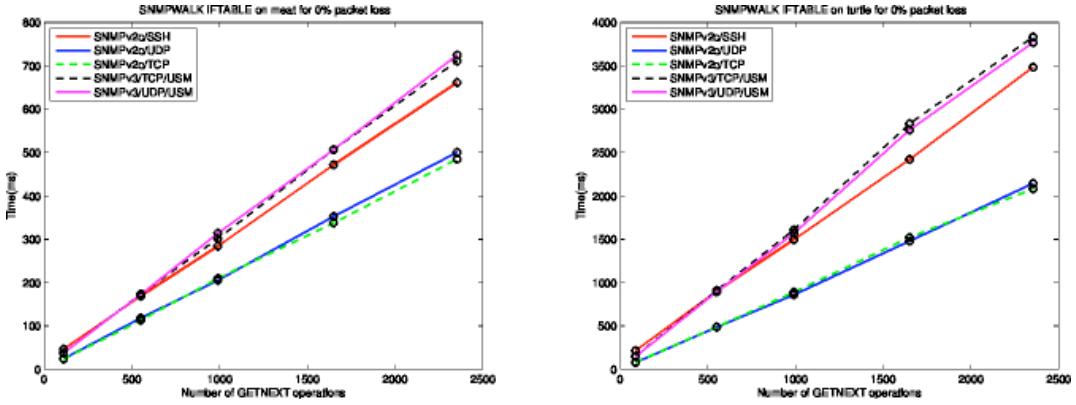

Figure. 4. 3. Latency comparison of SNMP getnext walks (ifTable) without packet loss with a command responder on a fast machine (left plot) and on a slow machine (right plot)

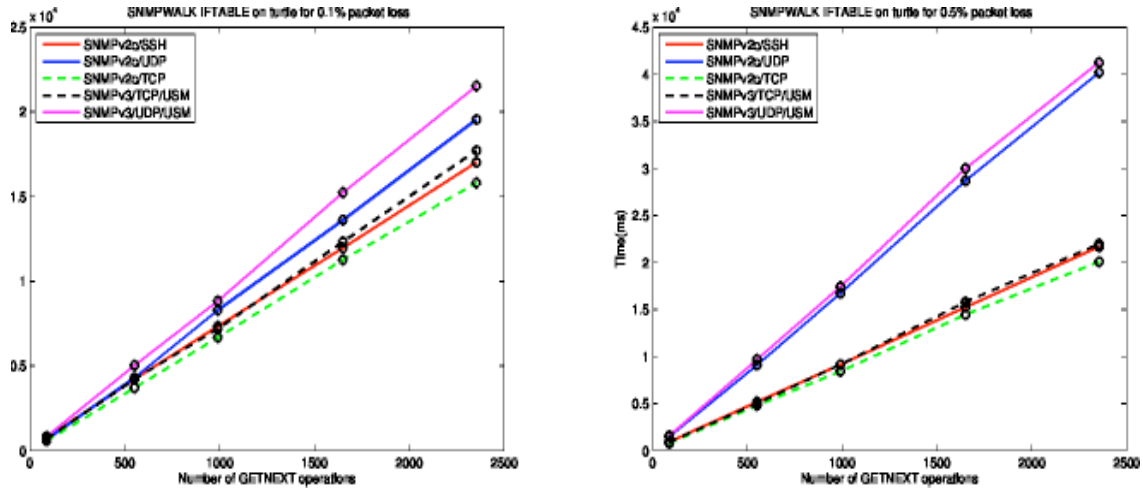

Figure 4.4. Latency comparison of SNMP getnext walks (ifTable) under packet loss with a command responder on a slow machine with different packet loss probabilities

Table 4.3 indicates that the change from SNMPv2c to SNMPv3/USM costs approximately 90 bytes of overhead per packet while the SNMPv2/SSH prototype adds approximately 40 bytes of overhead per packet to the SNMPv2c/TCP transport. While the SNMPv3 message header is slightly larger than the SNMPv2c header we used in our measurements, we believe that also an SNMPv3/SSH

Figure 4.4 shows the latency of the same snmpwalks on the ifTable with 0.1% and 0.5% packet loss using the slow command responder running on turtle. The surprising result is that the TCP-based transports all clearly outperform the UDP-based transports. It turns out that Net-SNMP has a very simple retransmission scheme with a default timeout of 1 second and 5 retries. TCP reacts much faster to lost segments in our experimental setup and this explains the bad performance of the UDP trans-

ports in the plots of Figure 4. Note that this result cannot be generalized since other SNMP implementations may have other retransmission schemes. However, simple statements that UDP transports outperform TCP transports in lossy networks are questionable as long as the application layer retransmission scheme is not spelled out.

Protocol	Range	Packets	Total Packets
SNMPv2c/UDP	80-100	4710	4710
SNMPv2c/TCP	110-	4709	4712
SNMPv3/USM/UD	170-	4710	4712
SNMPv3/USM/TC	200-	4709	4714
SNMPv2c/SSH	150-	4711	4742

Table 4.3. Packet sizes for the snmpwalk with 2354 getnext operations. The range column shows the dominating packet size range and the column packet the number of packets in that range; the column total packets indicates the total number of packets.

Figure 4.5 shows the amount of data exchanged during the snmpwalk experiments without packet loss. It can be seen that the amount of data exchanged increases when switching from UDP to TCP due to larger TCP headers. Furthermore, SNMPv2c/SSH requires less bandwidth than SNMPv3/USM/UDP and SNMPv3/USM/TCP. Even though the SSH connection protocol adds a tiny header, it seems that this header is significantly shorter than the space needed for the SNMPv3/USM header.

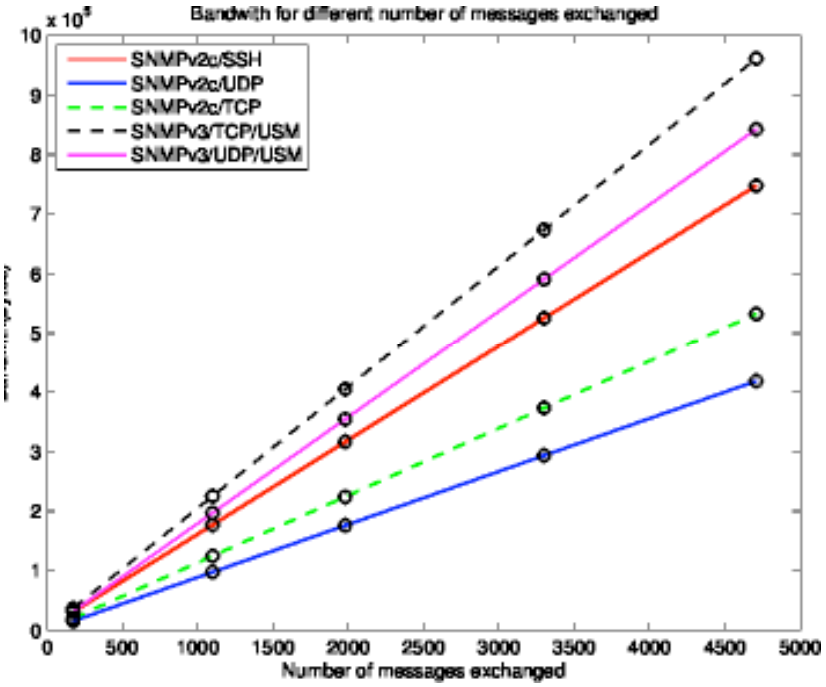

Figure. 4.5. Bandwidth used for different numbers of exchanged SNMP message

Figure 4.6 shows the amount of virtual memory used by the command responder process for an increasing number of concurrently open sessions. Initially, the process needed 9312 KByte of memory. The memory consumption growths linearly with the number of concurrently open sessions. The measured memory overhead per session is approximately 7 KByte. Note that the sessions were only established but not used

during these measurements. Some additional memory might be dynamically allocated by the SSH library once data exchanges take place.

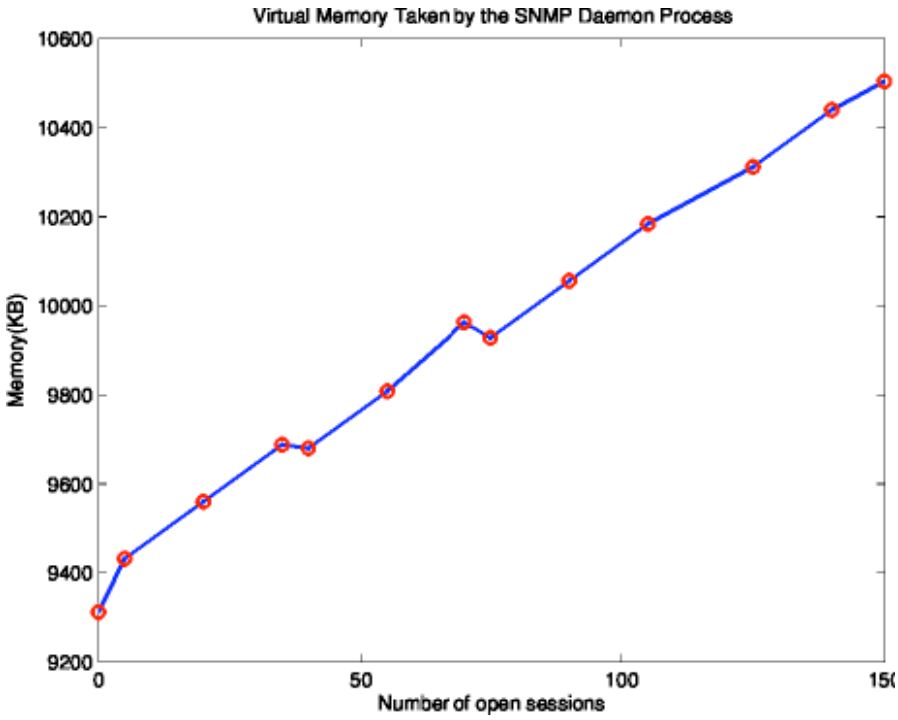

Figure 4.6. Virtual memory consumed by the command responder process for varying numbers of open sessions

5 Web services for network management - benchmarking

Web Services is an XML-based [12] technology that has attracted significant attention for building distributed Internet services. There have also been significant efforts trying to extend it to become a unifying management technology. An all-encompassing management technology needs to support efficient information retrieval, scalable event management, transaction support for configuration management and also security. Previous technologies, such as CMIP, SNMP and CORBA [13, 14] have addressed these aspects poorly, partially or at a high cost. This paper proposes an approach to address efficient information retrieval in terms of both bulk and selective data transfer. In order to achieve this, services modelling management information need to be organized in a hierarchy through service association. In order to achieve service association, information metadata are defined in secondary endpoints compared to the ones where services are deployed and accessed. We have defined a language for expressing arbitrarily complex information retrieval expressions and implemented a parser at the object level that evaluates these expressions, navigates arbitrary service associations and returns the results. We demonstrate the use and usefulness of the approach in an example usage scenario.

5.1 Service Design

Since the appearance of distributed object technologies, researchers realized the importance of converting SNMP and CMIS/P management information models. This led to the establishment of the Joint Inter Domain Management taskforce (JIDM) that produced a translation between the SNMP SMI / CMIS/P GDMO and CORBA's Interface definition Language (IDL). A JIDM-like translation though of SNMP SMI to services would be problematic for the same reasons encountered with its mapping to IDL. First bulk or selective retrieval is not supported. Second, scalability problems may arise when vast amounts of dynamic entities, i.e. SMI table rows, are modeled as separate services. As such, we considered an emerging approach for semantic translation of SNMP MIB's [3]. In such a translation, there may exist service methods for returning commonly accessed groups of single instanced objects, e.g. packet counters. In addition, tabular information is accessed collectively through a method that returns a table as a list of records. Additional methods may support SNMP-like "get next" or "get N next, i.e. get bulk" functionality. This type of modeling adds support for bulk data transfer for multiple instanced objects. Still, selective retrieval is not supported.

In WS bulk and selective information retrieval operations could be performed in two ways. The first method involves performing filtering actions at the SOAP [13] level with a tool such as the XML Path Language (XPath) [15] or similar. Since SOAP's body is in XML, XPath can be used to selectively pick portions of it. Such an approach is not problem-free though. Initially, extra latency is added in the retrieval process because more data need to be accessed, retrieved and coded in the SOAP body. Moreover, according to views expressed in the Network Configuration protocol mailing list, XPath is a very heavy tool for filtering purposes. Even a cut down version of XPath may still be resource intensive. A second approach to address selective retrieval is to perform it within the object code that represents a WS. As such we perform filtering before formation of the SOAP body, avoiding extra latency and keeping resource usage low by binding selective retrieval to the needs of the information model.

Supporting bulk retrieval for both single instance and multiple instance entities requires a collective view of management data across all levels of the data structure. To achieve this for SNMP, every MIB is modeled as a service. A level higher from the service level an agent has a collective view of all services, organized in a hierarchy through service association. The association scheme allows for arbitrary relationships to be defined. The agent in the scheme uses a parser to decide based on string expression queries it receives, the services from which data must be retrieved. Thus the agent has both a collective view and a selective capability over the services underneath it. At object code level, every service contains single instance and multiple instance entities of SNMP data modeled as simple values and tables. Bulk retrieval is achieved by three methods with different views on data. One method has access to single instance data, the other has view on table data and the third method has view of all the object data. All methods receive string arguments that represent command line queries that are interpreted by a parser we have built which decides which data will be sent to the manager requesting them. As such, all methods have both bulk and selective access to the data in their view. In Fig. 5.1 the translation of the SNMP's data into services is given. Modeling information this way initially allows a collective view of information at various levels (low level of granularity). At the same time a selective capability upon any data structure (service, simple data, tables) is offered (high level of granularity). Thus, our approach not only allows us to perform selective or bulk information retrieval but also to keep the number of services representing management data low. Therefore, it tries to avoid complexity and scalability problems.

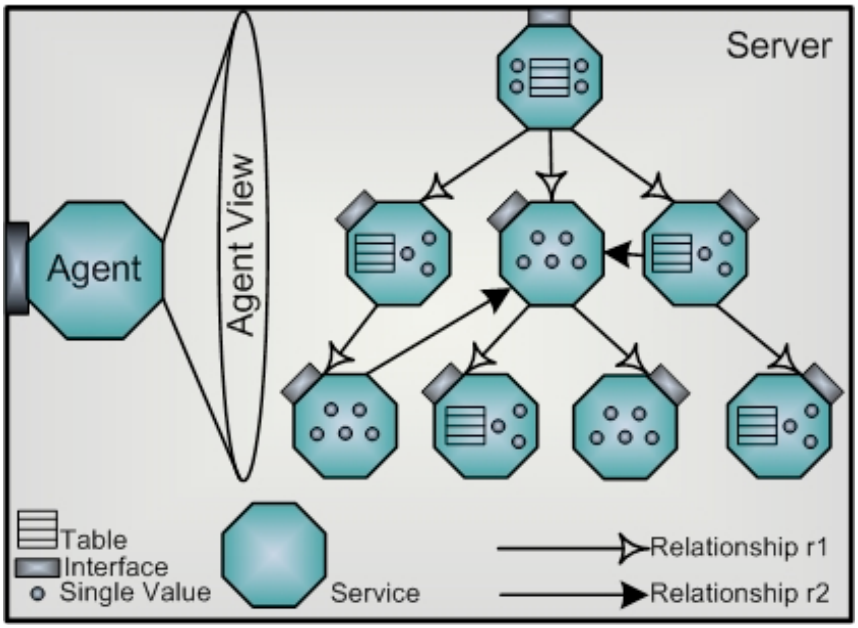

Figure 5.1. Modeling approach

5.2 Web-Services Association

From the information modeling approach presented in the previous section, it is evident that our scheme has two requirements. Firstly, it requires some means to define logical or physical relationships between services. These relationships will make navigation and selection of services feasible. These relationships provide an agent with a hierarchical view of the services underneath it and easy access to their data. A second requirement is that these relationships must be arbitrary so that traversal can be based on

any relationship. Our concept is to define different relationships between services and make navigation possible based on them. Services though have different access requirements than objects (i.e. more latency). Thus, we decided that it is more efficient to make service selection first and only then to apply selective actions on the data, in a similar fashion to what CMIS/P does with scoping and filtering. With these two requirements satisfied, our agent can selectively pick up services according to string expression queries it receives from a manager.
The common view for a hierarchical organization of entities is that of a tree, where elements in the previous level of the tree are connected with containment relationships with the ones in the next level. Navigation among the elements in this tree is based on containment. If these elements were services capturing SNMP MIBs and the relationships between these services were arbitrary, then a tree such as the one depicted in Fig. 5.2 can be defined.

Navigation of this tree and selection of services by an agent is possible in our scheme, by defining simple expressions that identify a starting point for the search, level restrictions for service selection and relationship patterns to follow. A simplified Backus Naur Form (BNF) syntax for such an expression is the following:

<path_selection_exp>::={< startpoint_tag> , <minlevel_tag> , <max-level_tag> , <pattern_tag>} . (**1**)

<pattern_tag>::=<identifier> | <pattern_tag> **.** <identifier> | (<pattern_tag>)! . (**2**)

<min_level_tag>::=<integer> . (**3**)

<max_level_tag>::=<integer> . (**4**)

<startpoint_tag>::=<identifier> . (**5**)

The path selection expression is dispatched from a manager to an agent in the form of a “command line” query expression. The agent uses it to select services based on relationship patterns, the level restrictions and the starting point tag. Selective retrieval of data from these services is performed only after selection of services has been completed. This is achieved by using the parser developed to interpret several other expressions that we will present later on. In order to demonstrate the usage of the path selection expression, some examples are given.

1) Path selection expression with no Restriction: A path selection expression such as the one in equation 6, if used with the information tree of services shown in Fig. 5.2, will cause a number of actions. Upon receiving the expression, the agent will use the parser to evaluate its validity. If the expression is valid then the agent will evaluate the expression and will start searching the sub-tree defined from the starting point (Root in this case) for services which can be reached by following relationships first of type r1 and then of type r2. The services selected are highlighted in Fig. 3. If selective retrieval expressions are also dispatched, the agent will only return the values that match the criteria posed by these expressions, as we will see later.

$$path\_sel\_\exp = \{Root,,,r1.r2\} \quad (6)$$

2) Path selection expression with single level Restriction: For the path selection expression in equation 7 the agent will start searching the sub-tree defined from the starting point (Root) for services in level 2 to which you can reach following relationships first of type r1 and then r2 . The selected services are highlighted in Fig. 5.4

$$path\_sel\_\exp = \{Root, 2, 2, r1.r2\} \quad (7)$$

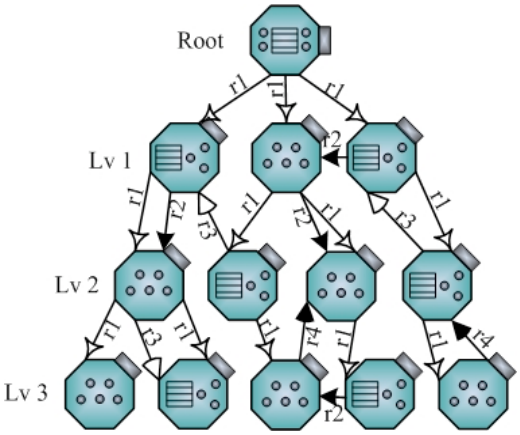

Figure. 5.2. General relationship tree

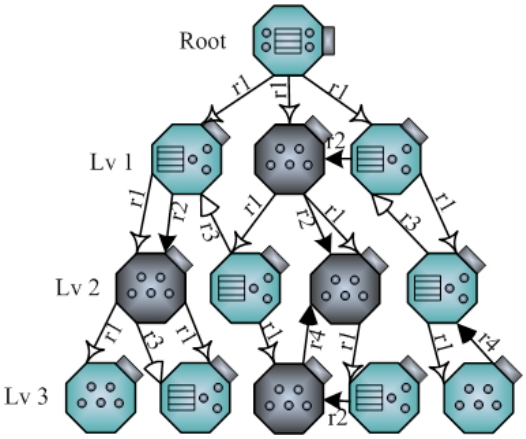

Figure. 5.3. Service selection no Restriction

3) Path selection expression with multiple level Restriction: In the case where the path selection expression has multi-level restriction, as in equation 8, the agent will search the sub-tree defined from the starting point (Root) for services in level 2 and 3 which can be reached by first following relationships of type r1 and then r2. The selected services are highlighted in Fig. 5

$$path\_sel\_\exp = \{Root, 2, 3, r1.r2\} \quad (8)$$

4) Fringe Services: In all the above service selection examples the agent visits one after the other all services included in the sub-tree whose head node is the start node tag. For every selection the agent makes, it evaluates for every service node whether each pattern tag in the relation pattern can be followed or not, thus there is a recursive evaluation of the binary state of each pattern tag in the relation pattern. The recursive evaluation of each pattern tag on a sequence of pattern tags can allow detection of services where the relation pattern cannot be followed. These services are called fringe services. An example of a path selection expression that captures services where type r1 relationships cannot be followed is given in equation 9.

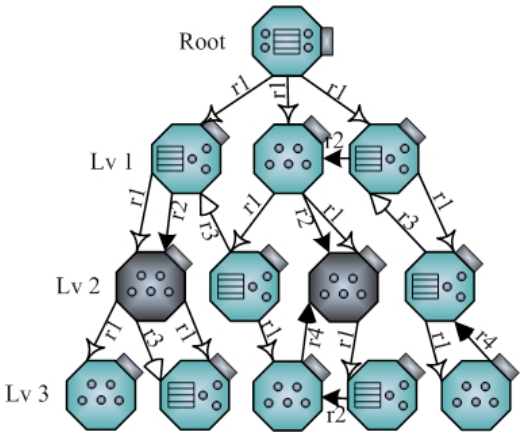

Figure. 5.4. Service selection single restriction

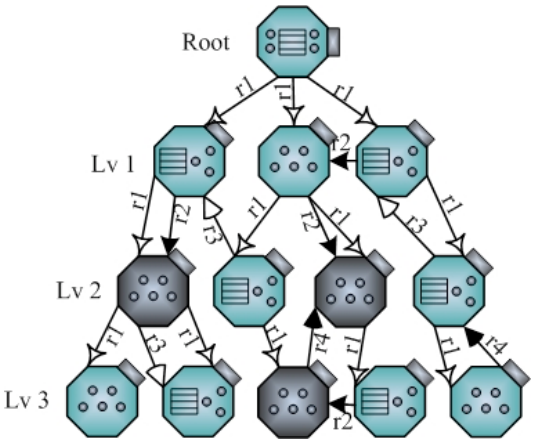

Figure 5.5. Service selection multiple restriction

$$path\_sel\_\exp = \{Root, , , (r1)!\} \quad (9)$$

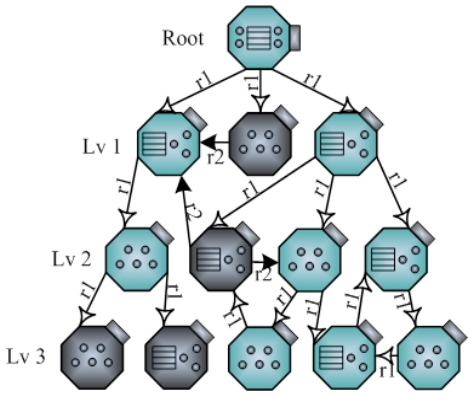

Fig.5. 6. Service selection for fringe Services

In order to use the selection scheme described previously, a method to define relations between services is required. A simple scheme to define such relationships if containment was the only option would be to use a simple naming scheme to define the endpoint URIs where services are deployed. Such a scheme would be to use the number of slashes "/" in the endpoint URI to denote the level in a hierarchy where a service is offered, the tag after the last slash to denote the name of the service and the tag before it to denote its parent. However, relationships between data may not only be containment. The definition of other relations must also be possible, so that the definition of actions between data and conversation scenarios between services is feasible.

One way to define relationships between services is to provide metadata about them. Initially we considered certain WS standards for the job. WS-Addressing [15] and WS-MetadataExchange (WS-MEX) [16] are such standards. WS-MEX specifies the messages that applications exchange in order to retrieve service metadata. WS-MEX is intended though as a retrieval mechanism for only service description data. Because introduction of metadata services will increase unnecessarily latency and memory requirements WS-Addressing was considered as another solution. WS-Addressing was initially designed to provide service endpoints with capabilities such as multi protocol and interface information support. This is achieved by adding service endpoint references (ER) to endpoint's Uniform Resource Identifiers (URIs). ERs are adding information to WS messages so that applications can support various exchange patterns than simple request responses. WS-addressing allows notification standards to provide asynchronous event reporting and subscription. It can be used though in a different way. WS-Addressing could be used to add information about service relationships. The addition of a metadata element in the standard to provide a consuming application with WSDL information also allows the provision of other metadata about a service. Still work on WS-Addressing is not finalized, while the standard and the metadata element is not supported by many open source toolkits. Thus, we had to find other means to support metadata information about service relationships until work in WS-addressing is finalized and open source toolkits support it.

Providing metadata about service relationships requires a simple and flexible scheme. In WS, WSDL allows to define the interface that services expose and the access point where they are offered. A WSDL document consists of an abstract part acting as an access stub and a concrete part affecting its behavior. The interface tag of the abstract part describes sequences of messages a service sends and/or receives. It does this by grouping related messages into operations. An operation is a sequence of input and output messages, and an interface is a set of operations [17]. The endpoint component of the concrete part defines the particulars of a specific endpoint where a service is ac-

cessed [17]. A service consists of its instance and its endpoint and the later is as important as the former. Most service properties and settings reside there.

The organization of WSDL and the structure it enforces on its constituent parts allows us to do three things. First, it allows manipulation of the level of transparency. Secondly, the granularity of services can be altered. Third it permits three distinct ways to deploy services. The most common way of deployment is by allowing access to an entire object through an interface. Service WS0 in Fig. 5.7 shows this deployment scenario. The WSDL document in this case contains one service tag referring to one binding and one endpoint tag. The second deployment scenario seen in Fig. 5.7 is for service WS1 where access to a service is through multiple access points. In this case the WSDL document for this service contains one service tag which includes multiple endpoint tags (two in this case) referring to the same binding tag. The third deployment scenario is seen for services WS2 and WS3. In this scenario two interfaces to the same object are offered by defining multiple endpoint elements for the same service element, which refer to different binding elements.

For our association scheme use of the second deployment scheme was made. The multiple access points of this scheme provide us with the means to define metadata about service relationships. In our proposal, every service has a primary access point to provide access to it. For every relationship a service has with another service, the latter will define a secondary URI. This secondary URI provides metadata about the relationship that the two services share with a syntax that complies with RFC 3986 [18] about URIs. The syntax for the primary and the secondary URIs is given in (10) and (11). Parsing secondary URIs provides agents with a view of the association tree.

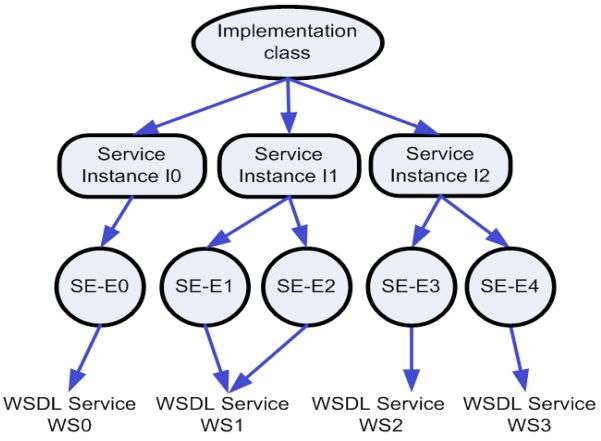

Figure. 5.7. Service deployment scenarios

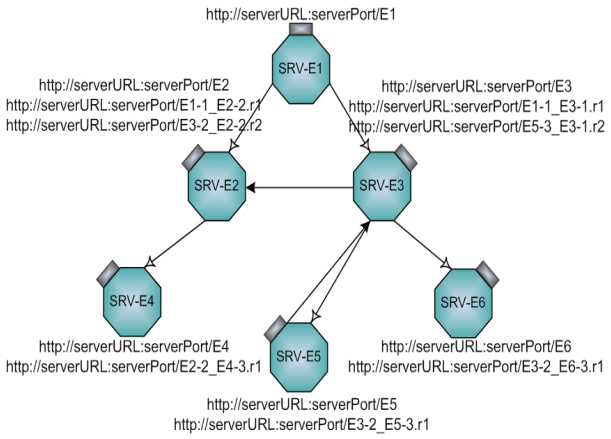

Figure. 5.8. Association scenario with endpoints

In Fig. 5.8 primary and secondary URIs for all services sharing two types of relations are provided. Service ST-E1 has only one URI to allow access to it. Services ST-E2 and ST-E3 have 3 URIs, the primary and two secondary ones for both the r1 and r2 types of relations they share with other services. All third level services contain one primary one secondary URI to show an association of type r1 with other services.

$$\text{Primary\_URI} = \text{http://serverURL:serverPort/primaryServiceTag} \quad (10)$$

$$\text{Secondary\_URI} = \text{http://serverURL:serverPort/sendingServiceTag-serviceLevel\_recipientServiceTag-serviceLevel.relationTag} \quad (11)$$

5.3 Selective retrieval

So far we have explained how bulk data retrieval is possible by providing a collective view upon data accessible by services at the agent level through service association and at the service level through specific methods that have collective access to the data. Path selection expressions allow our agent to deploy a selective capability on the services modeling the management information. To add filtering as selective retrieval capability on the management data within a service, our parser also evaluates data selection expressions sent to it by the manager. These expressions mandate which information from the service should be selected by the agent.

SNMP contains two types of data, single and multiple instance (tabular) ones. The BNF syntax for the data selection expression for single instance data is the following:

<sgl_slct_exp>::={<identifier > | <sgl_slct_exp >, <identifier> } . (**12**)

An example expression for retrieving the ipIndelivers, tcpOutSegs and the tcpInSegs from the TCP and IP MIB would be the following:

sgl_slct_exp={tcpInSegs,tcpOutSegs,ipInDelivers} . (**13**)

Retrieving multiple instance entities is a bit more complex since it requires an expression to define which entities need be retrieved and a filtering expression to retrieve only part of the data that meet specific criteria. The BNF syntax for the multi-instance selection expression and the filter expression is the following:

<mult_slct_exp>::={<mult_inst_tag> | <mult_slct_exp>, <mult_inst_tag >} . (**14**)

<mult_inst_tag>::=<identifier>([] | [<integer>-<integer>] | [< integer>] | [< integer>(< | >)><letter>(> |<)< integer>]) . (**15**)

<flt_exp>::={<mult_inst_tag ><relational operator> <value>| <flt_exp > <space><logical_operator><space> <flt_exp>} (**16**)

<value>::=<integer>|<string> (**17**)

A usage example for retrieving all TCP connections is given in (18).

ml_slct_exp={tcpConnEntry[ ]} (**18**)

The filter expression for retrieving only FTP and HTTP connections is given in (19).

flt_exp= {tcpConnLocalPort = 22 OR tcpConnLocalPort =80} (**19**)

5.4 Usage scenario

To demonstrate a case where fairly complex network management data must be retrieved, we present a use case scenario. In the configuration of Fig. 5.9, consisting of five IP routers and 6 Local Area Networks (LAN), LAN N2 receives substantial traffic from an HTTP server it hosts. Both N2 and R1 are able to cope with this traffic. At some point though, a user in LAN4 creates more traffic by transferring large files from an FTP server in LAN3. As a result R1 and LAN2 exceed their handling capacity. The cause of congestion in router R1 must be detected by the Central Management System (CMS) responsible for managing the overall network.

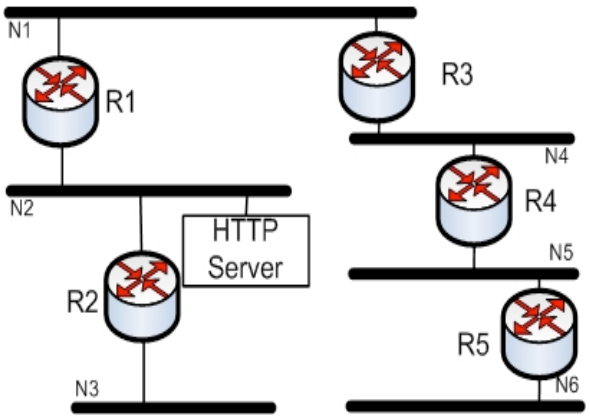

Figure. 5.9. LAN configuration

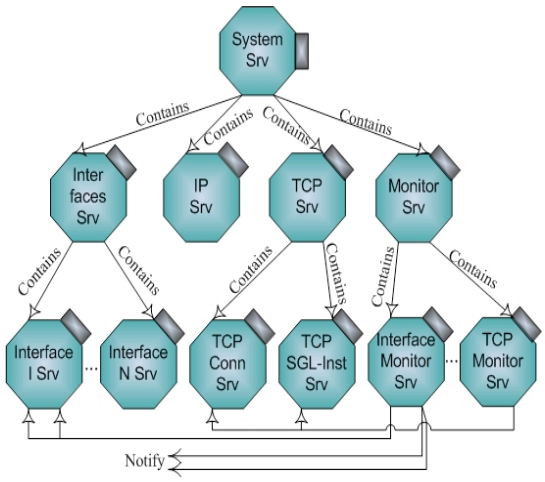

Figure. 5.10. Service association for usage scenario

Assuming the tree of services in Fig. 5.10 deployed in every host and router on the network, the CMS should be notified when congestion builds up and take appropriate actions. On notification the CMS must determine the cause of congestion and take actions to alleviate it. In Fig. 5.10, services IP, TCP, and Interfaces model the corresponding SNMP MIB groups. Other MIBs may also be captured as services but they are omitted. The TCP Service contains two services that model TCP single instance data and TCP connections. The Interfaces service breaks into a number of services representing the interface MIB data of every interface of the managed device. In Fig. 5.10, other generic services such as an event service or a logging service might also be present. To keep things simple we have omitted a WS-based notification service and assume that notifications to the CMS are sent from the services that produce them. In the future we will investigate creation of a flexible and scalable WS-based notification service. At the moment when the traffic congestion threshold is crossed on interface N2 the relevant interface monitoring service notifies the CMS; we assume here that the CMS has activated the interface monitoring service in R1 Upon receiving this notification, the CMS must try to determine the cause of congestion.

The CMS should retrieve information about the load that TCP connections between the various subnetworks are imposing on the interface N2 of router R1. Such information is not provided by SNMP MIBs. Such an option would be available, if MIBs would be capable to capture incoming and outgoing traffic per TCP connection or use the RMON MIB to capture link layer traffic. In our case, let's assume that the TCP MIB is equipped with such data in the TCP connection table under two variables called tcpConnInSegs and tcpConnOutSegs. In this case, the CMS should start monitoring hosts with Web or FTP servers and try to identify TCP connections with high segment counters and also high segment rates.

Path_slct_exp={TCP Monitor Srv,NULL,NULL,empty} (**20**)

flt_exp={ (tcpConnLocalPort = 22 OR tcpConnLocalPort =80) AND
(tcpConnInSegs>thres_value OR tcpConnOutSegs>thres_value) (**21**)

For the CMS to inform the agent which data it requires to retrieve from the TCP monitoring service, it should dispatch to the agent the expressions in (20) and (21). On receiving these two expressions, the agent picks up from the TCP monitoring service only TCP connections for applications on well known ports, usually known to produce traffic. Such applications are FTP and HTTP on ports 22 and 80 respectively. In addition, the

filter expression tells the agent to only retrieve connections whose incoming or outgoing traffic exceeds a certain threshold. This way the manager will acquire a few possible candidates responsible for creating congestion. With further monitoring on these connections, it can determine the behavior of each one in terms of segment rate and possibly identify remote hosts through tcpConnRemAddress that cause the extra traffic. Without the functionality we support, the whole of the remote TCP connection table would need to be retrieved, and this would incur a lot of additional traffic to the already congested network. This is a relatively simple scenario but other scenarios also exist where a lot more information that belongs to different services must be selectively retrieved. One such case may be the selective retrieval of data from the logging service in order to trace particular series of events.

6 JMX performance evaluation

To define management delay metrics for JMX, we start from the IPPM framework [19] for IP packets delays measurements. We identify a monitoring delay metric to be relative to a monitoring attribute defined by its name, its delay metric type, the management operation type used to achieve the management task, the agents identifer and the number of management objects carried by the operation. For example, the delay to retrieve the FreeMemory attribute from a single agent to a manager is represented as following:

FreeMemory-AttributeDelay-GetAttribute-Manager-Agent-1.

We identify two types of delay metrics: the attribute delay and the group delay. The attribute delay is relative to a single monitoring attribute retrieved from a single agent. This metric is for interest to capture the delay to move an attribute value between two management nodes (for example from an agent to a manager). The group delay captures the delay to move an attribute value from a group of agents with a fxed size. This metric is sensitive to perturbation that might arise on retrieving one attribute or computing an aggregated attribute from a group of agents.

The attribute delay is the total delay that an attribute experiences to retrieve/alter its values with a specifc interaction mode (polling or notifcation), using a specifc management operation. The attribute delay is measured in a time unit (seconds or milliseconds) per target attribute. If the interaction mode is notification, then it is a one way delay experienced from a sender to a receiver. If the interaction mode is polling, it is the round trip time between two or more management nodes.

The group delay is the amount of time required for a management algorithm to retrieve/alter the value of an attribute from a group of agents. The group delay is indeed the longest delay served, during a polling round, from any agent of the group. The unit of measure of the group delay is time units per agents group size. Many management algorithms are sensitive to this amount of time since it impacts the algorithm reaction time. During a management task period and according to its scheduling operations approach (sequential/serial or concurrent/parallel) [14], the group delay is dedned by the maximum of attribute delays from the agent's group in a parallel management operations scheduling approach, or the sum of the attribute delays in a serial approach.

There are three approaches to delay measurement . One is to collect packet-level information somewhere in a network path between the manager and agents. Another is to collect kernel-level information on the manager side. The third is to collect information at the application level on a manager or a mid-level managers side. The packet and the kernel level approaches provide the most detailed information and the most accurate timings. Application-level measurements are easy to implement, and the benchmarking tools are portable. We use application-level measurement to develop and evaluate management delays. The benefit of application-level measurement, in contrast to other levels measurement, is that it reflects delays as perceived by a decision point implemented on a manager or a mid-level manager sides. Thus, the measured delays will include network delays, requests and responses processing delays on each management node, matching and searching attributes values delays on the agent, and security and privacy processing. Figure 6.1 depicts a time series of measured delays on the manager side at the application-level in a Java-based management framework. We observe some periodic spikes where delay increases due to garbage collector activities each 60 seconds. Therefore, only an application-level measurement technique captures easily these spikes that might affect the performance of the man-

agement algorithm. The disadvantages of the application-level measurement are the overhead that introduces when logging measurements datasets, and the loss of information about each individual component delays among agent delays, manager delays and messages delays.

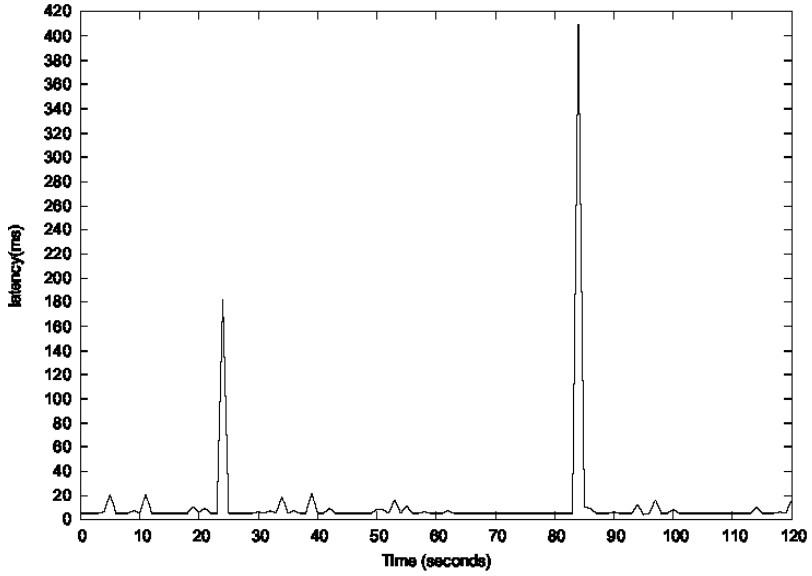

Figure 6.1. A time series of measured delays at the application-level on a manager side while sending 1 request per second to a single agent over JMX framework.

The data we present and analyse in this deliverable was collected from our JMX benchmarking platform [20] running on a cluster of 100PC (I-Cluser2) [1] where nodes are connected via a gigabyte Ethernet. We built a synthetic benchmarking application based on some widely used JMX implementations (SUN Reference implementation 1.2 and MX4J 2.0). The synthetic application is used because it provides flexibility in experimentation with various levels of workload: number of requests per second, type of requests, type of MBeans and the number of attributes on each registered MBean within the MBean server. We used a BEA WebLogic JRockit JVM from a JDK 1.4.2 04 running on an Itanium 2 with 2x900MHZ CPU and 3GB memory. We kept the default options values of all running JVMs on nodes. For all experiments we fxed the monitoring rate in terms of number of requests per second and we varied the number of agents. On the manager side, we used a concurrent monitoring operations scheduling strategy. The manager creates for each agent a pool of threads that generates a fxed number of requests per second. We use the JMX *getAttribute* operation that carries out a single attribute value from a single MBean. Thus, the size of requests is the same for all experiments. For each agent we stored its requests delays in a separate le, during a measurement of 20 minutes after a warm-up period of 1 minute as recommended in [21]. Within the measurement period, we record the timestamps before and after calling the *getAttribute* operation on the manager side. The difference between the two timestamps represents the measured attribute delay. We use the Java instruction *currentTimemillis* to get the current time in milliseconds with a resolution of 1 ms on Linux operating system[2]. All measurement data are collected and stored on a separate node. Confronted with 20GB of collected data to analyse, it is clear that we cannot hope to individually analyse each trace. We must indeed turn to *automated analysis.* That is, we use developed Perl scripts to analyse our data and generate the corresponding statistical properties.

4.3 Statistical analysis techniques

The first analysis technique we applied is summarizing a data set. If we wish to summarize the attribute delays we might at first think to express them in terms of their sample mean and variance (or standard deviation). However, in practice we find that often the collect of an attribute experiences one or more delays that are much higher than the reminder. These extreme delays greatly skew the sample mean and the variance, so that the resulting summaries do not accurately describe a typical behavior.

A typical management delays description is reflected by the median and the IQR (Inter-Quantile-Range) that remain resilient in the presence of extremes or outliers.

This typical description is suitable for management algorithms that require unbounded delay to retrieve management data. However, an extreme description is reflected by the mean and the standard deviation since they are less robust to outliers. Hence, the choice of statistical estimators for summarizing delays datasets depends on the evaluated management algorithm quality requirements. For a real time monitoring algorithm, the mean and the standard deviation are more adequate to summarize monitoring delays since they reflect any perturbations that might occur on the monitoring system. One other technique we apply, is describing delays using statistical distributions of collected data. The empirical distribution function (EDF) [19] of a set of scalar measurements is a function F(x) which for any x gives the fractional proportion of the total measurements that were ~ x. If x is less than the minimum value observed, then F(x) is 0. If it is greater or equal to the maximum value observed, the F(x) is 1. Moreover, in our analysis we match the group and attribute delays data sets against the popular Weibull distribution model [22] to approximate the underlying statistics.

6.1 Attribute delays analysis

Firstly, we analyse the attribute delay when transferring the values of the same attribute from a single agent within a variable group size of agents to a manager under a fixed monitoring rate of 1 request per second. We know that the performance bottleneck is located at the manager side since the management approach is centralized. Table 6.1 shows the attribute delay statistics measured on the manager side when retrieving values of an attribute from an arbitrary agent within a group.

Group size	Attribute Delay (ms) statistics					
	Mean	Std	Median	IQR	Min	Max
70	1.02	0.20	1	0	1	4
140	4.45	46.34	1	0	1	973
210	68.47	432.37	1	0	1	3614
280	105.08	983.41	1	0	1	9960
350	392.73	2250.64	1	0	1	20269
420	675.24	3689.65	1	0	1	33365
490	561.55	4254.14	1	0	1	39816
560	1787.82	8179.39	1	0	1	50395
630	1916.86	8764.2	1	0	1	60646
700	2394.55	12073.7	1	0	1	78994

Table 6.1. Summary of attribute delays statistics measured at the mangaer side from a single agent within a varied group size.

We observe from table 6.1 that the attribute delay description varies according to the used statistics estimators as mentioned in the section 6.3. When increasing the size of the group of agents, we observe first that the minimum delay is constant, the mean and the standard deviation increases. Indeed, the median and the IQR remain invariant and constant.

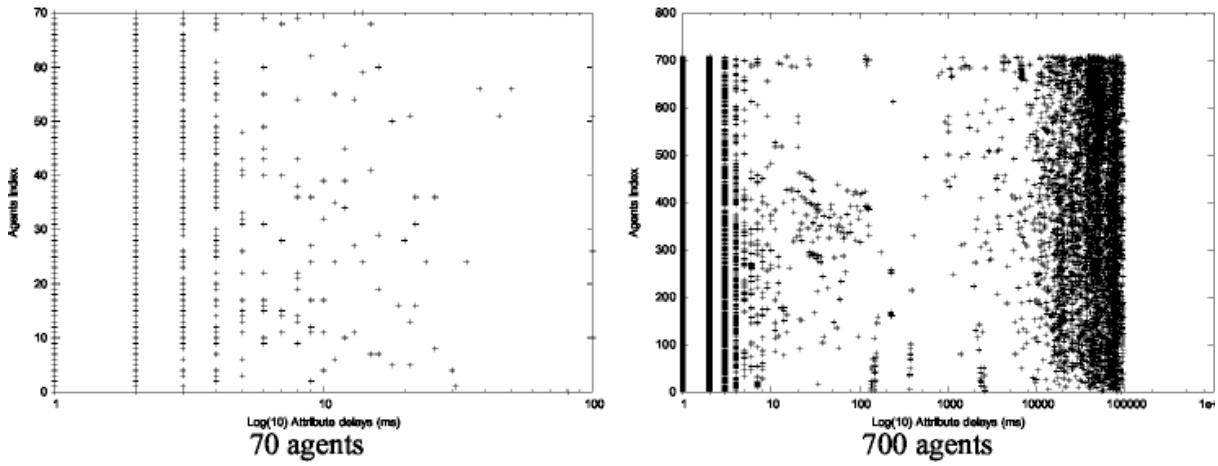

Figure 6.2. Empirical time series of attribute delays per agent from two groups different sizes.

Figure 6.2 shows the time series of attribute delays of different agents from groups of size 70 and 700. We observe that the attribute delays become more randomness while the group size increases. We then consider an associated random process $X = \{Xi\}i=1,2,..m,$ where Xi is the attribute delay of the *ith* agent within a group of size m. We are interested in the behavior of a random variable $\{Xi\}$ within a group of size m. Figure 6.3 shows the empirical distribution of an attribute delay from an agent within a group size of 700.

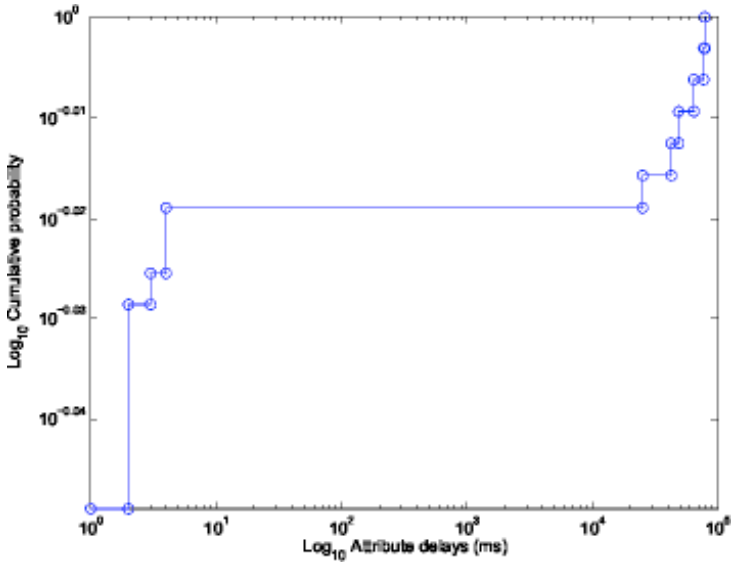

Figure 6.3. Log-Log plot of the Empirical Distribution of the attribute delays of an agent within a group size of 700.

We did not find a suitable statistical distribution fitting these delays. Indeed, we observe that the distribution of attribute delays from an agent is multimodal as depicted in the Figure 6.3. The lower 95% of the data are shorter delays that closely follow a normal distribution *N*(1.10,0.44). The 5% of the data at higher delays follows closely a Weibull distribution as depicted in Figure 6.4.

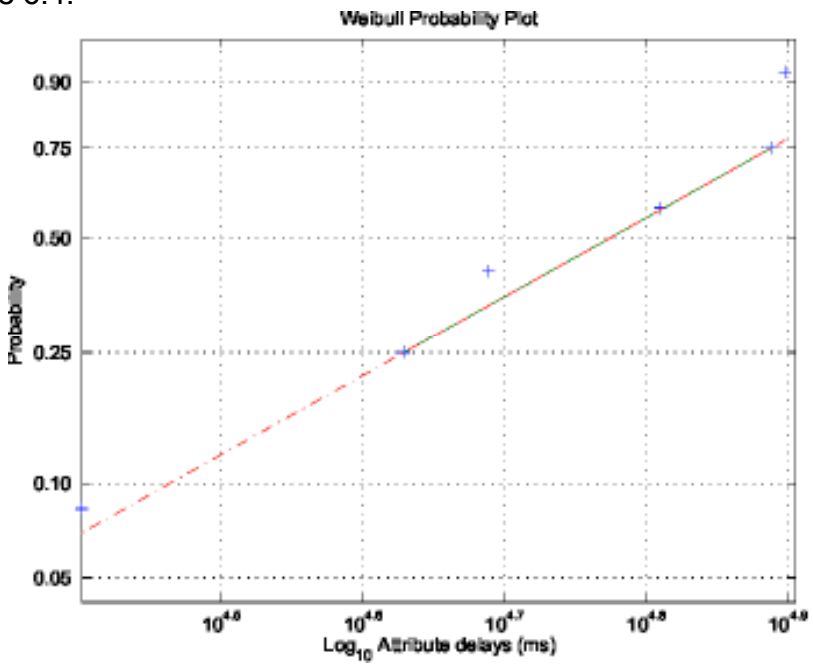

Figure 6.4. Weibull plot of the upper 5% of attribute delays

6.2 Group delay analysis

Secondly, we analyse the group delay of an attribute from a varied size group during a management period (a polling round for example) with a given management task (monitoring or configuration) and using a given management operation. Within a management period, the group delay is defined as a random process X = {Xi}i=1,2,,..m of a group of agents with size m. In this section, we are interested in the behavior of X. We plot the ECDF function as depicted in figure 6.5 of the mean attribute delays from each agent with different group sizes varying from 70 to 700. The group mean attribute delays shift right when the group size increases that means the monitoring messages are more burstiness and the queueing delays increase on the manager side. We use the maximum likelihood estimators to find an adequate classical statistical underlying distribution of group delays with a fixed size. Our finding is that the Weibull distribution best fits the group delays data set. This distribution has been recognized as a good model for TCP inter-arrival times and for many fields in engineering reliability components lifetimes.

Figure 6.6 reports the Weibull probability plot of the group attribute mean delays distribution of 140 agents, with respect to the best fitted Weibull distribution over the same data set. The parameters a, b of the Weibull distribution represent the so called scale and shape parameters. When the shape parameter is set to 1, the Weibull distribution degenerates into an exponential distribution. When it is smaller than 1, the tail of the distribution tends to be heavier, while for values larger than 1 the shape of the distribution tends to assume a Dumbbell form.

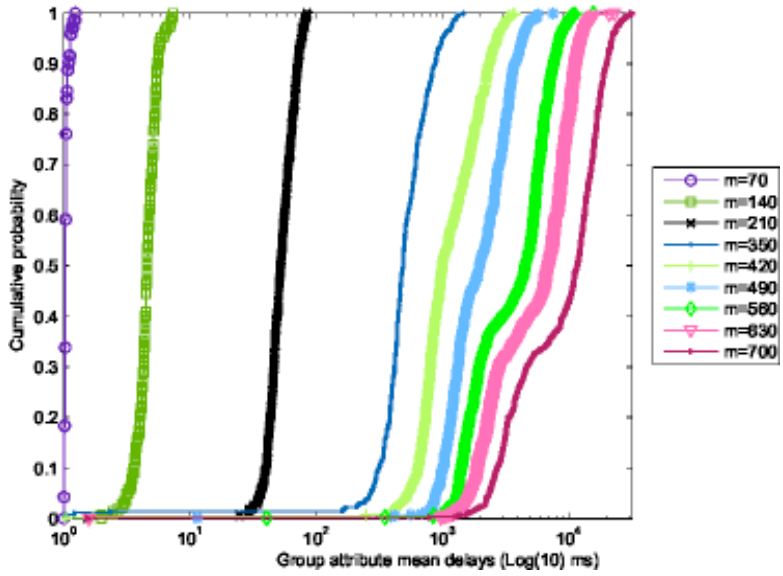

Figure 6.5. Empirical Distribution of the group attribute mean delays of a monitoring task with a monitoring period of 1 second, using a polling interaction between one manager and a group of agents with size *m,* and using the JMX GetAttribute operation.

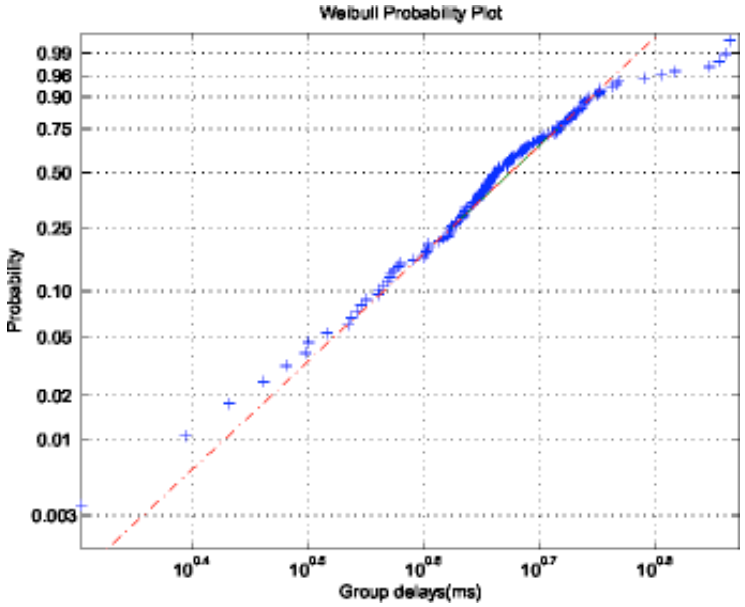

Figure 6.6. The Weibull probability plot of group attribute mean delays of an agent's group size of 140. The fitted Weibull distribution has a scale parameter *a* = 5.02 and a shape parameter *b* = 5.52.

The classical maximum likelihood method was used to obtain the best *a* and *b* parameters for the fitting procedure. We observe that as the shape parameter decreases, the probability of longer as shorter attribute mean delays values increase while the burstiness of the group size increases. If the shape parameter is close to 1, the Weibull distribution approximates an exponential distribution and the group attribute mean delays becomes more random. The variation of the Weibull parameters from the size of the group of agents *m,* bounds the degree of freedom of developed models parameters to predict management

delays distributions of similar random variables derived from data sets other than the ones used in this work. But our aim is that our model predicts the general shape of future distributions but not the exact form. If the distribution parameters are known for a future dataset, then the model becomes fully predictive for that dataset.

6.3 Management delays quality

To gauge how well management delays scale when the size of agent's group increases, we are interested to identify the proportion of attribute delays that are less or equal to a maximum tolerable delay that we consider, in this work, as the monitoring interval D (1 second in our case). Let q be this proportion of attribute delays, statistically we obtain q = P[X ~ D] = EDF(D). The quantity 1 - q represents the proportion of attribute delays that are late after the monitoring interval and q is a quality parameter of a monitoring framework delay. For the group delay metric, the parameter q becomes more interesting, since it captures the proportion of agents that respond within a delay less or equal to the monitoring interval. This quantity represents the monitoring error of the management system and captures the quality of the monitoring algorithm temporal accuracy. Table 2 shows the measured and predicted values from the fitted distribution of the 1 - q parameter.

Group size *(m)*	Measured monitoring error 1-q	Predicted monitoring error 1-q	Weibull parameters	
70	0	0	17.11	1.07
140	0	0	5.52	5.02
210	0	0	4.54	59.16
280	0	0	1.81	77.54
350	0.05	0.04	2.28	610.36
420	0.51	0.62	1.93	1468.88
490	0.91	0.86	2.06	2519.79
560	0.99	0.95	1.79	4902.03
630	0.99	0.98	1.75	7437.27
700	0.99	0.98	1.63	12349

Table 6.2. The measured and the predicted monitoring error 1-q of group delays based on on the attribute mean delays of agents.

6.4 Cross-validation

In order, to give a first insight on the validity of our model, we use the group mean attribute delays dataset from the study of Pras et al [24] to partially verify that group delays approximate a Weibull distribution. The validation is partial because their datasets are measured for SNMP based monitoring system and are packet-level measurements. We focus on the dataset delays of retrieving a single object from the group of agents. Our finding that their group mean attribute delays of their data set with a group size of 23 agents fits closely a Weibull distribution as depicted in the figure 7. In their paper, the authors record SNMP agents Round-Trip delays at the UDP-level, indeed our dataset delay are recorded on the application-level with RMI as underlying protocol that lies on the TCP protocol. As noted in [23], wired TCP flow arrivals are well modeled by a 2-parameters Weibull distribution and claimed that it gives a better fit than other distribution models (lognormal, pareto and exponential). Hence, according to these works

based on packet-level measurement and our work based on application-level measurement, we claim that the Weibull model is a good candidate to model delays in management frameworks.

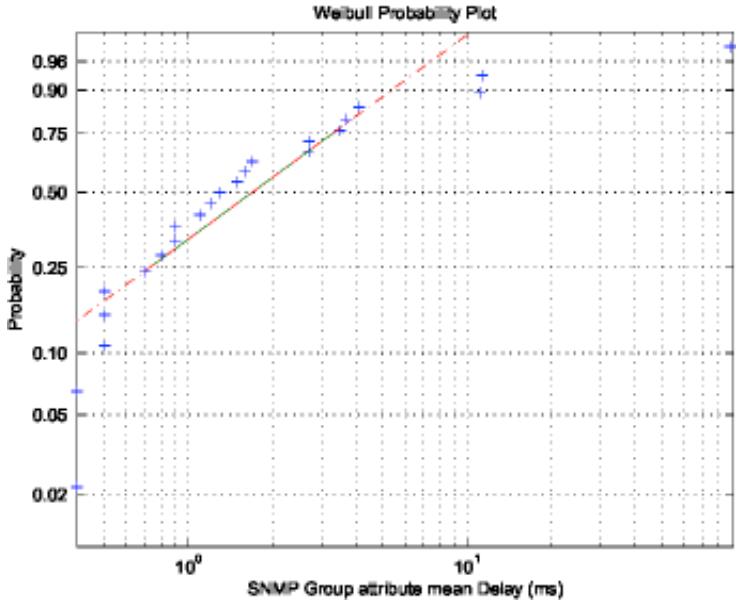

Figure 6. 7. The Weibull Probability plot of group attribute mean delays of SNMP agent's group size of 23. The fitted Weibull distribution has a scale parameter *a* = 3.46 and a shape parameter *b* = 0.62.

7 Summary and Conclusions

In this deliverable we presented an approach to address the benchmarking of different management frameworks. We studied a broad range of management protocols and for each of them, we performed extensive tests.

The first study considered the case of the emerging IETF endorsed NetConf protocol. We addressed the security and filtering mechanisms that are possible and performed various benchmarking tests. A second management framework is related to another XML based framework centered on the usage of web services for efficient retrieval of management data.

Viewing management information collectively at various levels of the management hierarchy addressed the problem of bulk retrieval. Using a parser to interpret expressions that highlight only specific data to be retrieved solves the problem of selective retrieval.

In order to support collective view of data at the object level, we deployed methods that view SNMP single instance or multiple instance data as a whole. To do the same at the service level we developed a scheme that defines arbitrary relationships between services. In order to allow the navigation and selection of services based on any relationship we developed a parser that interprets appropriate expressions. In order to perform selective retrieval at the object level, the same parser interprets another series expressions to highlight the data that will be selected.

We will be investigating a proper notification service in the near future, tracking also work that has taken place in relevant standards bodies.

A third major area of research concerned the performance of SSH enhanced SNMP. SSH is already widely used to secure the access to command line interfaces on network elements. Accordingly, SSH credentials (keys, passwords) are readily available in many environments. This availability of credentials in many operational networks has been the main motivation for considering SSH as a secure transport for SNMP. The ISMS working group of the IETF is working on new security models for SNMP which leverage a secure transport. One crucial question is how the performance of this new approach compares to the existing security solution for SNMP (SNMPv3/USM) and to the still widely deployed insecure versions of SNMP (SNMPv1/SNMPv2c).

The measurements presented in this deliverable try to give answers to some of these questions. In particular, we quantified the session establishment overhead for SNMP over SSH. For simple one-shot SNMP requests, SSH seems to be a rather costly solution since the costs for establishing a session and associated session keys is significant. For sessions that carry multiple SNMP interactions (e.g., table walks), the costs for the initial session setup are amortized and there is a break-even-point where SNMP over SSH starts to become more efficient than SNMPv3/USM with authentication and privacy enabled. The answer to the question whether SNMP over SSH is a viable alternative to SNMPv3/USM therefore depends on the SNMP usage pattern and the typical session length. While SNMP traditionally has no concept of a session, it is possible to approximate session life times by analyzing SNMP traffic traces. Work is underway in the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF) to collect SNMP traffic traces from different operational networks. These traces are expected to give insights in which environments SNMP over SSH is likely to be a viable alternative.

The fourth major framework that was considered is JMX, a management framework suited for service and application management. Many investigations have studied partially management delays as part of their proposed management frameworks, but to our knowledge none of them did investigate the statistical properties of these delays and how they matched with well known underlying statistical distributions. The main used statistics to summarize management delays was the mean and the standard deviation. Most of these works assume that management delays and specifically monitoring delays are uniform. Such an assumption is heavy and becomes invalid when both the size of the number of managed resources and management agents increase. In these cases, monitoring algorithms do not scale well from a delay perspective where they loose the desired quality (timeliness and temporal accuracy).

In this deliverable, we have analysed management delays within a simple centralized monitoring algorithm. Our primary objective is to identify a set of metrics that characterize delays. We identified how to use the statistical estimators (mean, standard deviation or median and IQR) to characterize management delays based on the quality requirements (delays dependant or independent) of the evaluated algorithm. Our analysis of a synthetic JMX based management benchmark datasets, shows that the group delays has a statistical underlying distribution, identified as the Weibull model. Understanding the statistical modelling of management delays is important for the simulation and analytical performance evaluation studies for management frameworks. It is also interesting for the proper designing of management applications (buffer sizing and underlying transport protocol). Our main objective in this work is to initiate empirically derived analytical models of management frameworks, that could be exploited in simulation environments or performance prediction. As a further interesting work, we will investigate the coupling of packet-level and application-level delays measurement datasets to better understand management frameworks delays and develop more accurate models.

8 References

[1] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. XML-Signature Syntax and Processing. W3C Recommendation, February 2002.

[2] T. Imamura, B. Dillaway, and E. Simon. XML Encryption Syntax and Processing. W3C Recommendation, December 2002

[3] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson, December 2002.

[4] U. Blumenthal and B. Wijnen. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414, Lucent Technologies, December 2002.

[5] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, SSH Communications Security Corp, Cisco Systems, January 2006

[6] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, Enterasys Networks, BMC Software, Lucent Technologies, December 2002.

[7] D. Harrington and J. Sch¨onw¨alder. Transport Mapping Security Model (TMSM) Architectural Extension for the Simple Network Management Protocol (SNMP). Internet Draft draft-ietf-isms-tmsm-02.txt, Futurewei Technologies, International University Bremen, May 2006.

[8] D. Harrington. Secure Shell Security Model for SNMP. Internet Draft draft-ietf isms-secshell-02.txt, Futurewei Technologies, March 2006.

[9] A. G. Morgan. The Linux-PAM Application Developers' Guide. Technical report, November 1999.

[10] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, Cisco Systems, Argon Networks, June 2000.

[11] R. Presuhn. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). RFC 3418, BMC Software, December 2002.

[12] W3C, "The Simple Object Access Protocol 1.2 (SOAP)", http://www.w3.org/TR/soap12-part1,h ttp://www.w3.org/TR/soap12-part2

[13] W3C,"The Web Services Description Language 1.1 (WSDL)", http://www.w3.org/TR/wsdl/

[14] G. Pavlou et al, "CMIS/P++: extensions to CMIS/P for increased expressiveness and efficiency in the manipulation of management information.," 7th Annual joint conference of the IEEE Computer and Comms Societies, INFOCOM 98, Vol 2, April 1998 ,pp.430 – 438

[15] D. Box, F. Curbera, "WS-Addressing specification" , W3C submission 10 August 2004.

[16] F. Curbera, J. Schlimmer, "WS-Metadata Exchange specification", Sept. 2004.

[17] OASIS, "The Universal Discovery Description and Integration Technical Committee Draft V 3.02 (UDDI)," http://uddi.org/pubs/uddi_v3.html.

[18] T. Berners-Lee, R. Fielding, L. Masinter, "The Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[19] Paxson, V., Almes, G., Mahdavi, J., Mathis, M.: RFC 2330: Framework for IP performance metrics (1998) Status: INFORMATIONAL

[20] Andrey, L., Lahmadi, A., Delove, J.: A jmx benchmark. Technical Report RR-5598, LoriaINRIA Lorraine (2005)

[21] Demarey, C., Harbonnier, G., Rouvoy, R., Merle, P.: Benchmarking the round-trip latency of various java-based middleware platforms. Studia Informatica Universalis Regular Issue **4** (2005) 7–24 ISBN: 2-912590-31-0.

[22] Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions. 2 edn. Volume 1. John Wiley & Sons (1994) ISBN: 0-471-58495-9.

[23] A.Feldmann: Characteristics of TCP connections. In: Self-similar Network Traffic and Performance Evaluation. Jonhn Wiley and Sons (2000) 367–399

[24] Pras, A., Drevers, T., de Meent, R.V., Quartel, D.: Comparing the performance of SNMP and web services-based management. eTransactions on Network and Service Management (eTNSM) **1** (2004)

[25] Lahmadi, A., Andrey, L., Festor, O.: On the impact of management on the performance of a managed system: A jmx-based management case study. In: DSOM. Volume 3775 of Lecture Notes in Computer Science., Springer (2005) 24–35

[26] Lahmadi, A., Andrey, L., Festor :On Delays in Management Frameworks: Metrics, Models and Analysis, Proc of the 17 th IEEE/IFIP DSOM 2006.

[27] Marinov V., Schönwälder J *:* Performance Analysis of SNMP over SSH**,** Proc of the 17 th IEEE/IFIP DSOM 2006.

[28] Chourmouziadis A, Pavlou, G. Efficient Information Retrieval in Network Management Using Web Services. Proc of the 17 th IEEE/IFIP DSOM 2006.

[29] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, Y. Yemini - "Decentralizing Control and Intelligence in Network Management", Proceedings of the 4th International Symposium on Integrated Network Management, Santa Barbara, CA, May 1995

[30] Pere Vilà, Josep L. Marzo, Institut d'Informàtica i Aplicacions (IiiA), Universitat de Girona - "Scalability study and distributed simulations of an ATM network management system based on intelligent agents"

Abbreviations

CMS	Content Management System
CSS2	Cascading Style Sheets 2
DMTF	Distributed Management Task Force
GNU	The GNU Project
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IRTF	Internet Research task Force
IPPM	IP Performance Metrics
ISO	International Standards Organization
ITU	International Telecommunication Union
JMX	Java Management eXtensions
MANET	Mobile Ad-hoc Networks
NOC	Network Operations Center
NoE	Network of Excellence
OMG	Object Management Group
PHP	PHP Hypertext Preprocessor
QoS	Quality of Service
RFC	Request For Comment
RSS	Really Simple Syndication
SNMP	Simple Network Management Protocol
TMF	Tele Management Forum
TMN	Telecommunication Management Network
W3C	World Wide Web Consortium
WS	Web-Services
XML	Extended Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

9 Acknowledgement

This deliverable was made possible due to the large and open help of the Work Package 7 team of the EMANICS NoE. Many thanks to all of them.